ADR™ DEVELOPMENT STANDARD

ADR™ Commands and Logical Tape Format for Driver Development

Driver Development Description for DI-30, DP-30, SC-30, and SC-50 OnStream ADR™ Drives

Version 1.1

Preliminary

Important Notices

Information furnished in this manual is believed to be accurate and reliable. OnStream, Inc. reserves the right to change ADR development standards at any time without notice. Applications described in this document for any of these products are for illustrative purposes only and are provided as is. OnStream, Inc. makes no warranty for such applications, nor for any outcome from their usage. OnStream, Inc. assumes no responsibility for any errors that may appear in this document.

OnStream, Inc. assumes no responsibility for the use of this document; nor for any infringements of patents or other rights of third parties which may result from its use.

ALDC - represents Adaptive Lossless Data Compression, which is the property of IBM Corporation. ALDC is the algorithm prescribed for use in hardware data compression by this standard.

Overview of Revision Changes:

Rev. 1.1.001

- Initial Draft 001, adapted from ADR Logical Format specification 1.0

Rev. 1.1.002

- Draft 002
- Remove buffer management host speed control sections.
- Fix code example math for center load tapes.

Rev. 1.1.003

- Draft 003
- Corrected read algorithm to avoid IDE bus hang potential during read error recovery.
- Fixed code example for creating header frame, there was an error in the tape capacity calculation.

Rev. 1.1

- Removed Draft status, established as a Preliminary release document.
- Added device identification section in chapter 2.

Rev. 1.2

- Corrected placement of header frames, second set moved to 0xBAE through 0xBB2 (see Page 11 section 3.2.1). The second set of header frames had been erroneously placed in an area that was previously reserved. Failure to move the second set of header frames to 0xBAE through 0xBB2 has been shown to cause problems with some existing software.
- Bumped minor revision (section 3.2.2.3) to indicate that this is version 1.2.

TABLE OF CONTENTS

LIST OF TABLES

List of Figures

1. Introduction

1.1 Introduction

This Development Standard specifies the logical format requirements for data recorded on an ADR ™ tape device. Specifically, this document addresses the use of the DI-30 (IDE/ATAPI), DP-30 (Parallel Port), and SC-30/50 (SCSI) models of the drive. It is highly recommended that any driver conforming to this standard check the device inquiry string and only activate for the device models listed above.

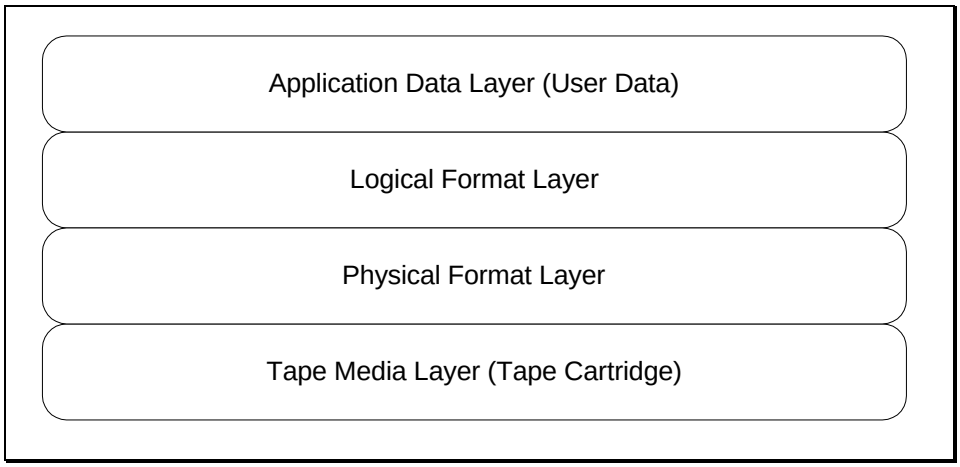

Figure 1-1 Layers within the ADR Format

The logical format layer resides between the user data recorded by the application and the physical format layer.

1.2 Definitions

For the purpose of this Standard, the following definitions apply:
Appendable Point: An appendable point is where a write operation is allowed to begin.
AUX Data: The AUX Data is a 512 byte control field associated with each 32 KB tape frame.
BOLG: BOLG indicates the Beginning of a Logical Group. The BOLG is defined in the Data Access Table Entry.
BOP: The Beginning of Partition, which is used to signal the first tape frame in a partition.
BOT: The Beginning of Tape, which is the take-up reel end of the tape.
Configuration Frames: The Configuration Frames are reserved non-user data frames that are used for configuration and directory information. The Configuration Frames are located in the Configuration Partition, which is the first partition located at BOT.
COT: The Center of Tape.
ECC: ECC represents an Error Correction Code that is used for correcting recoverable data.
EOD: The End of Data, which is used to mark the end of the valid data area for a partition in a streaming tape application.
EOLG: EOLG indicates the End of a Logical Group. The EOLG is defined in the Data Access Table Entry.
EOP: The End of Partition, which is the last tape frame of a partition.
EOT: The End of Tape, which is the supply reel end of the tape.
Filemark: A virtual point within the sequence of data blocks placed by the host to aid navigation. A read operation will terminate upon reaching a Filemark. Each Filemark is assigned a unique logical block address.
Kbytes (KB): This Standard defines 1 KB to be equal to 1024 bytes.
Logical Block: A Logical Block is a group of data bytes that the host application specifies.
Logical Element: A Logical Element represents an entity on tape that possess a unique logical block address. This specification supports two types of logical elements, data blocks, and filemarks.

Logical Group: A logical group is the sequential grouping of logical elements that possess the same attributes and fit into a single tape frame.
Logical Track: A group of physical tracks recorded or read simultaneously through a multi-channel head forming a single logical entity.
Load Point: The point at which the drive will position the tape when it is first inserted.
LP: Load Point.
Magnetic Tape Cartridge: A cartridge containing magnetic tape wound on two coplanar hubs with an internal drive belt to transport the tape between the hubs, the two hubs form a supply reel and a take-up reel for the tape.
Parking Zone: A region of tape where the tape may be stopped prior to ejecting the tape where no user data is at risk due to exposed media.
Partition: A sequence of logical data blocks with logical addresses starting from zero. All tapes have a minimum of one partition labeled partition 0.
Reserved (res): Reserved fields are to be written with zeros and ignored by firmware to facilitate future enhancements.
Tape Frame: A complete data entity consisting of 32 KB of user available data combined with ECC data and control fields. Each tape frame shares a one-to-one correspondence with a particular physical location on tape and is synchronized to the buried servo.
Track: A longitudinal area on tape along which magnetic signals are serially recorded. Each track is marked by a buried servo signal, and positioned between BOT and EOT.
Undefined: Undefined fields may contain any value. The contents of an undefined field are not considered to have any meaning. These may be fields which are inactive in a certain mode or are reserved for future enhancements, but which are not required to be zero filled.

1.3 Byte and Code Requirements

Byte: A group of 8 data bits operated on as a unit.
Byte Length. The data shall be in eight-bit bytes. The 8 bits in each byte are numbered b0 to b7, b7 being the most significant bit.
Byte Order. Big endian ordering shall used for variables larger than one byte, all efforts shall be made to ensure that independent variables are placed on even address boundaries (word length).
Text Code. Bits b0 to b6 correspond to the 7 least significant bit assignments specified in the American National Standard Code for Information Interchange (ASCII), ANSI X3.4 - 1986. To comply with this Standard, bit 7 shall always be set to Zero and the seven bits b0 through b6 shall represent ASCII characters.
Radix. All numbers are decimal, base 10, unless followed by an 'h' suffix or a 0x prefix that indicates hexadecimal, base 16. A 'b' suffix indicates a binary bit field.
Word: A group of 16 data bits operated on as a unit.
Word Length. The data shall be in sixteen-bit words. The 16 bits in each word are numbered b0 to b15, b15 being the most significant bit.
Double-Word Length. The data shall be in thirty-two-bit double-words. The 32 bits in each byte are numbered b0 to b31, b31 being the most significant bit.
Signed Numbers: Negative numbers are represented by the 2's complement.

2. ADR Data Format Overview

2.1 Device Model for DI-30, DP-30, SC-30, and SC-50

In order to provide the cost/capacity level achieved in the DI, DP, and SC models of the ADR drive the device intelligence has been simplified. The command interface provides a mechanism for locating to physical frames on tape, reading frames sequentially, and writing frames sequentially. The tracking of the logical elements on tape is left to the application software or drivers. The functionality is in many ways similar to the traditional “floppy” tape drives.

2.2 Device Identification

In order to minimize compatibility problems with future devices, it is recommended that applications implementing the device model described in this manual only activate when one of the following model strings appears in the data returned from an Inquiry command:

OnStream Model	Inquiry Data Bytes 16-32 (ASCII characters)															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DI-30	‘D’	‘I’	‘-’	‘3’	‘0’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’
DP-30	‘D’	‘P’	‘-’	‘3’	‘0’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’
SC-30	‘S’	‘C’	‘-’	‘3’	‘0’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’
SC-50	‘S’	‘C’	‘-’	‘5’	‘0’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’	‘ ’

Table 2-1 Inquiry String Model Identification

2.3 Use Model

To standardize the nature of the data recorded on tape, this document prescribes a use model for recording logical elements on tape including logical data blocks and filemarks. Strict adherence to this description will provide the maximum possible level of interchange with current and future ADR products. Many of the data structures are designed to support much more complex issues than are apparent in this use model; this is to allow future implementations to include features like variable blocks and multiple partitions. In order to bound the complexities in implementing a standard streaming device emulation, and in keeping with the limited command set available, the use model described here is limited to the following subset of features:

- User data blocks are fixed blocks 32 KB in size, one logical block per tape frame
- A 512 byte control field must also be written with each 32 KB frame for a total 32.5 KB transfer
- The entire tape consists of one partition (excluding reserved configuration frame locations)
- An append only streaming model is implemented

Data structures are provided for the frame control fields and for the data area in the reserved media management frames (called configuration frames). Algorithms are provided for write and read operation including defect management and drive buffer management issues.

The drive is functionally capable of an update-in-place operation. The application can locate to a particular frame address and update only that frame without effecting the adjacent frames. This mode of operation is inconsistent with the standard streaming tape model, and as such, it is only used for managing configuration frames in this use model.

2.4 Tape Frames

The ADR is a multi-channel, serpentine recording device that implements a buried servo system for accurate track following. Multiple physical tracks form a logical track. The data is recorded along linear tracks that run between BOT to EOT. Data is physically organized into frames with each frame interleaved across the physical data channels. All frames are synchronized to the buried servo pattern and all frames are of the same length and interval as measured from the buried servo signal. All logical tracks on a tape have the same number servo signal cycles and thus all logical tracks on a tape have a fixed number of frames. Each frame location is assigned a unique tape frame address.

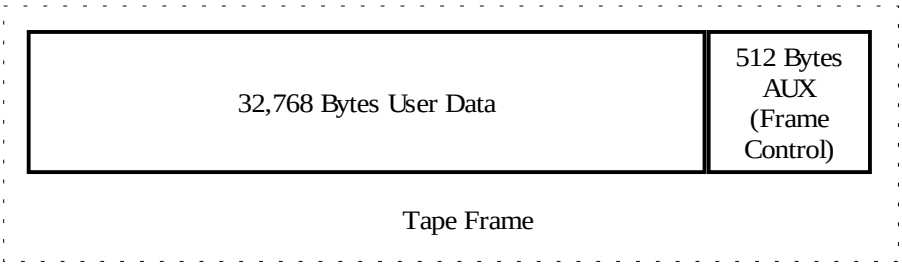

Figure 2-1 ADR Tape Frame as seen by application software

A tape frame is the physical element that is recorded on the tape. The frame consists of 32,768 (32K) of user data as well as 512 bytes of control field (known as the AUX) and multi-level ECC parities. All ECC generation, verification, and restoration are handled internally in the drive hardware and are not visible outside the drive.

3. Logical Data Format

3.1 AUX Data

The principal tool for implementing the logical format is the AUX Data. Each tape frame contains 32 Kbytes of data storage space plus 512 bytes AUX Data. The AUX Data is set aside as a control field area for use by the device in order to navigate and catalog the information recorded in each tape frame.

3.1.1 AUX Data Locations

The AUX Data describes the physical location, the type of elements, and a description of the elements contained within a tape frame. The following table defines the AUX Data bytes (all reserved fields must be zero filled):

Position	Bytes	Usage
0-3	4	Format Identifier (0x00000000)
4-7	4	Application Signature
8-11	4	Hardware Field (0x00000000)
12-15	4	Update Frame Counter
16-17	2	Frame Type
18-19	2	Reserved
20-35	16	Partition Description
36-43	8	Reserved
44-47	4	Frame Sequence Number
48-55	8	Logical Block Address
56-187	132	Data Access Table
188-191	4	Reserved
192-195	4	Filemark Count
196-199	4	0xFFFFFFFF
200-203	4	Last Mark Frame Address
204-223	20	Reserved
224-255	32	Driver Unique
256-511	256	Reserved

Figure 3-1 AUX Data Structure

3.1.2 Format Identifier

This field indicates what level of internal drive hardware compatibility the AUX is based on, it shall always be 0 for the DI-30, DP-30, SC-30, and SC-50 drives.

3.1.3 Application Signature

All applications must identify themselves to drive before writing to the tape is enabled. The Application Signature must be initialized through a vendor unique Mode Select page, once initialized the drive will place the signature in each frame written. The signature is a four-character ASCII string unique to each application.

3.1.4 Hardware Field

The Hardware Field is reserved for use by the formatter ASIC, the application should treat this field as reserved on writes, on reads this field is undefined.

3.1.5 Update Frame Counter

The Update Frame Counter is only used in configuration frames, in all other frames this field shall always be zero filled. In configuration frames, such as the header frame, this counter is incremented each time the data in one of those frames is updated. Since there are several redundant copies of each configuration

frame there is the possibility of conflicting copies of the frame in question. The frames with the highest number contain the most recent information. When the tape is first written with configuration frames, this field shall be written with 0x00000000. This field shall be incremented by one each time the data in the frame in question is updated. The count is sufficiently large that a tape should exceed life before this counter reaches saturation.

3.1.6 Frame Type

The Frame Type field is contained in the AUX data bytes 0 and 1. The Frame Type is a two (2) byte bit field that identifies the frame, see Figure 3-2. The following frame types are described in section 3.1.6.1 through section 3.1.6.3.2.

- Header Frame
- Filler Frame
- Marker Frame
- Data Frame
- EOD Frame

The Frame Type bit definitions are provide in Figure 3-2 while the corresponding value for each frame type is given in Table 3-1.

byte\bit	7	6	5	4	3	2	1	0
0	Data	res	res	res	Header	res	Marker	EOD
1	res	res	res	res	res	res	res	res

Figure 3-2 Frame Type Field

Frame Type	Byte 0 Value	Byte 1 Value
Filler Frame	0x00	0x00
EOD Frame	0x01	0x00
Marker Frame	0x02	0x00
Header Frame	0x08	0x00
Data Frame	0x80	0x00

Table 3-1 Frame Type Values

3.1.6.1 Configuration Frame Types

The Configuration Frames are located at the load point of the tape (beginning of the tape, or at the center of tape on those cartridges where recording begins at a center point). The Configuration Frames shall not contain user's data. For additional information about the Configuration Frame data structures, see section 3.1.10.5.

The Configuration Frame types are:

- Header Frame

3.1.6.1.1 Header Frame

The header contains basic information identifying the nature of the data on tape. It defines any partitioning and provides tables to improve random positioning times.

3.1.6.2 Universal Frame Types

The universal frame types are those that may exist in any area of the tape and do not contain user data. The following universal frame types are defined:

- Filler Frame
- Reserved Frame Types

The universal frames shall only support the Frame Type fields in the AUX Data, all other AUX Data fields shall be undefined.

3.1.6.2.1 Filler Frame

Filler frames contain the vendor unique information only. All data in a filler frame shall be vendor unique and non-critical. Filler frames will most often contain no meaningful data, but they may be used as a volatile working space.

3.1.6.2.2 Reserved Frame Types

Any frame type not defined in this specification is reserved for future enhancements. Any frame type that is not recognized by a device shall be treated as a filler frame.

3.1.6.3 Logical Element Frame Types

The logical element frame types contain elements that are placed on tape based directly on data or commands from the host. Logical element frame types shall only exist in data partitions, no logical element frame types shall be recorded in the configuration area. The logical element frame types defined are:

- Marker Frame
- Data Frame
- EOD Frame

3.1.6.3.1 Marker Frames

Marker Frames are used to indicate Filemarks. Each Marker Frame shall contain one mark. Filemarks are a logical entity that shall have a unique logical block address. Filemarks are only used in the Streaming tape model. As a logical entity, the Filemarks are cataloged in the Data Access Table table for the Marker Frame.

3.1.6.3.2 Data Frames

Data frames contain space for 32 Kbytes of user's data.

3.1.6.3.3 EOD Frame

Though not a true logical element, EOD is used in conjunction with logical elements in a streaming tape model. EOD indicates the end of data in a partition. An EOD frame is used to identify the append point for subsequent write operations. An EOD Frame shall contain the EOD data structure. The EOD data structure is used to validate the accuracy of the current Track Table.

3.1.7 Partition Description

The Partition Description is a data structure defining the nature of the current partition. The structure identifies the partition, defines the boundaries of the partition, and provides a data validity check in the form of a write pass counter.
The Partition Description is recorded in the AUX field of each frame in the partition and is a member of the Partition List structure in the Header Frame.

Byte	Description
0	Partition Number
1	Partition Description Version (0x01)
2-3	Write Pass Counter
4-7	First Frame Address
8-11	Last Frame Address
12-15	reserved

Figure 3-3 Partition Description

3.1.7.1 Partition Number

This is the partition identification, each partition shall have a unique partition number. The partitions that are currently defined in this specification are –1 (0xFF), and 0 (0x00). Partition –1 is the "configuration

partition" which is the area of reserved frames near the load point that are reserved for media management structures.

3.1.7.2 Write Pass Counter

The write pass counter is implemented in all streaming tape model operations. The write pass counter is changed on each write from BOP within a partition during a streaming tape model write. During streaming tape model operations data, filemarks, and EOD marks are only considered current and valid if they contain the current value in the write pass counter. The write pass counter value shall only be modified during the creation of a new partition, the logical erasure of a partition, or an over-write of a partition from BOP. When a new write pass counter value is required it is assigned as the Write Pass Counter plus one. The Write Pass Counters for each partition are maintained in the Partition Description data structure.
Old data may exist in a partition from a previous write if during a streaming overwrite operation a servo tracking fault condition arises. The purpose of the write pass counter is to assure that during a subsequent read operation any old data that is detected will not be considered to be part of the current data stream.

The Configuration Partition (partition –1) is not a streaming tape partition, and as such, this field has no meaning in the Configuration Partition. In the Configuration Partition this field shall always be –1 (0xFFFFFFFF).

3.1.7.3 First Frame Address

This field defines the location of the beginning of the partition.

3.1.7.4 Last Frame Address

This field defines the location of the end of the partition.

3.1.8 Frame Sequence Number

The Frame Sequence Number is a four-byte counter that starts from zero in a partition and increments by one with each valid frame that is in a user data partition. The Frame Sequence Number in combination with the Write Pass Counter in the Partition Description is used to determine the validity of the data during reads.
During a write operation, a number of frame locations may be skipped over due to a write error. A subsequent read operation over the area can detect that such an event occurred by seeing a discontinuity in Frame Sequence Numbers and/or an old Write Pass Counter. By reading further down the tape the application can resynchronize the read operation when it encounters the valid write pass count and finds the expected sequence number.
Note: In the use model described in this document all logical blocks are 32KB, the same size as the user data area of a tape frame. The result is that this counter will actually turn out to be equal to the Logical Block Address that is in the next field.

3.1.9 Logical Block Address

The Logical Block Address is four words wide. The Logical Block Address defines the frame's logical position within a partition. A Logical Block Address is assigned to any frame that contains valid Filemarks or user data.
The Logical Block Address represents the logical block address for first logical block start contained in the tape frame. In the case of very large logical blocks, where no logical block start exists in the tape frame, the logical block address of the block in progress is used. The Logical Block Address begins from zero at BOP and shall increase monotonically towards EOP. The Logical Block Address is assigned to the first logical element that contains a BOLG in the tape frame. BOLG is defined in the Data Access Table, see section 3.1.10.3 for additional information.

3.1.10 Data Access Table and Logical Block Descriptions

The Data Access Table shall be supported in the streaming tape mode. The Data Access Table provides the means for determining the nature and location of the logical elements within a frame.

3.1.10.1 Logical Elements on tape

A Logical Element represents an entity on tape that possesses a unique logical block address. This specification supports two types of logical elements. Each logical element on tape has a unique logical address within a partition. The logical block address always starts from 0 at the BOP and the address sequentially increases for each logical element written in sequential order.

- Data Blocks
- Filemarks

3.1.10.2 Logical Groups

A logical group is the sequential grouping of like logical elements that differ only in logical block number.

3.1.10.3 Data Access Table Format

The Data Access Table describes the logical elements or partial logical elements contained in the tape frame. When a format discontinuity or a tape frame boundary is encountered, or when a compression group is described, a new Data Access Table entry is made. A Data Access Table entry shall only describe the components of the logical groups contained in the current tape frame or the current compression group.

Byte	Description
0	Data Access Table Entry Size (8)
1	reserved
2	Number of Data Access Table Entries (n)
3	reserved
4-131	**(n) Data Access Table Entries**

Figure 3-4 Data Access Table Structure

3.1.10.3.1 Data Access Table Entry Size

Each Data Access Table Entry is 8 bytes long.

3.1.10.3.2 Number of Data Access Table Entries

This is number of entries in the Data Access Table, which corresponds to the number of format discontinuities and compression group descriptions contained in the frame.
The Data Access Table supports up to 16 entries. No more than 16 format discontinuities and compression group descriptions shall reside in a single tape frame. The Data Access Table entries beyond those that are enumerated contain undefined data.
Note: Some loss of capacity will be experienced in the event that the Data Access Table is filled before the tape frame is filled with user data.

3.1.10.3.3 Data Access Table Entries

Each discontinuity in attributes within a frame shall require a unique Data Access Table entry. Each compression group shall require two Data Access Table entries.

Byte	Description
0-3	Logical Block Size
4-5	Number of Logical Elements
6	Data Access Table Entry Flags
7	reserved

Figure 3-5 Data Access Table Entry Structure

3.1.10.3.3.1 Logical Block Size

A logical block represents the data block size used across the host interface. This format specification does not require that the logical block size have any particular relation to the tape frame size, nor to any of the lower level physical elements defined in the physical tape format document.
The Logical Block Size (LBS) indicates the number of bytes contained in one logical block. If a logical block spans a tape frame boundary, then the Logical Block Size shall contain the number of bytes actually

in the current frame. To determine the host logical block size of logical blocks that span frames, the device will have to process all of frames containing the components that make up the complete logical element
In a marker frame the Logical Block Size shall always be zero.

3.1.10.3.3.2 Number of Logical Elements

The Number of Logical Elements (NLE) indicates the number of logical elements or partial elements described in the Data Access Table entry. For a partial element Number of Logical Elements shall be set to one, since one element is being partially described, this is the case when spanning a tape frame.

3.1.10.3.3.3 Data Access Table Entry Flags

The Data Access Table Entry Flags are used to indicate the attributes of the logical element described by the Data Access Table entry.

Bits	7	6	5	4	3	2	1	0
Description	EXT	CMP	res		BOLG	EOLG	res	MARK

Figure 3-6 Data Access Table Entry Flags

CMP: If this bit is set to one (1), then the logical element(s) within the logical group is compressed data. If this bit is set to zero (0), then the logical element(s) is uncompressed data. This compression refers only to ALDC hardware data compression. Drivers written to operate with SC, DI, and DP family drives should reject the medium as an incompatible format upon an attempt to read a tape frame with this bit set. Drivers written to operate with SC, DI, and DP family drives should always write a zero (0) to this bit.
BOLG: The Beginning of a Logical Group bit indicates that the beginning of this logical group is contained in this frame. Drivers complying with this specification shall always record logical blocks that are 32KB, and are always contained within a single frame, so the BOLG bit shall always be set to one (1).
EOLG: The End of a Logical Group bit indicates that the end of this logical group is contained in this frame. Drivers complying with this specification shall always record logical blocks that are 32KB, and are always contained within a single frame, so the EOLG bit shall always be set to one (1).
EXT: The Extended Entry bit indicates that the next entry in the Data Access Table is additional information concerning this logical group. This is only used with ALDC hardware compressed data, the first entry in the Data Access Table describes the compression group while the extended entry describes the compression group contents. Drivers written to operate with SC, DI, and DP family drives should reject the medium as an incompatible format upon an attempt to read a tape frame with this bit set. Drivers written to operate with SC, DI, and DP family drives should always write a zero (0) to this bit.
MARK: In a data frame the MARK bit shall always be set to zero (0). For a Data Access Table entry in a marker frame a Filemark is represented with the MARK bit set to one (1) (a MARK bit setting of zero (0) in a marker frame is reserved). Each marker frame shall contain one (1) tape mark. The "Number of Logical Elements" field must contain a 1 in a marker frame.

Logical Block Size: The "Logical Block Size" indicates the number of bytes contained in one logical block. In a Marker frame the Logical Block Size shall be set to zero (0).
Number of Logical Elements: The "Number of Logical Elements" indicates the number of logical elements in the Data Access Table entry.

3.1.10.4 Data Compression

This specification supports embedded hardware data compression as an optional feature for the streaming tape model. The compression algorithm shall be the ALDC algorithm. For compatibility, the only onboard hardware data compression algorithm is ALDC.
Other external data compression algorithms that occur before data is passed to the drive are beyond the scope of this specification and have no effect on the AUX data.

3.1.10.5 Filemarks

This specification supports Filemarks. Since Filemarks represent logical elements and format discontinuities, an entry in the Data Access Table is required for a filemark. Each filemark shall have a unique logical block address. All filemarks shall be recorded in a Marker Frame, which separates filemarks from user's data. A Marker Frame shall contain one (1) filemark and shall not contain user data.

The filemark counters are designed to optimize scanning for filemarks in the reverse direction possible by constantly maintaining a reference to the previous filemark location. From any data frame or marker frame the application can locate directly to the previous marker frame.

3.1.11 Filemark Count

The Filemark Count field is a running count of filemarks in the partition up but not including this tape frame.

3.1.12 Last Mark Frame Address

The Last Mark Frame Address contains the frame address of the last tape mark that was recorded in the current partition. If no tape marks have been written in the current partition then this field shall be filled with 0xFFFFFFFF.

3.1.13 Driver Unique

The Driver Unique area is a 32 byte region allocated for the optional use and definition by the application. It may be used as a scratch area, or it may be used for optimization features that are tuned for a specific application. It should not be made mandatory for basic operation as different applications may use it differently.

3.2 Configuration Frames Data Structures

Configuration frames are stored in redundant copies near the load point of the tape. The configuration frame locations for the Header Frame and the Defect Map Frame are specified in the ADR physical format.

3.2.1 Configuration Frame Locations

Configuration Frames are located at fixed physical block numbers in redundant copies, repeated on two tracks. When updating configuration frames, all copies of the frame must be updated. Locations are given as a range of Frame Numbers. Locations are provided for two reserved regions. There are five redundant copies of each configuration frame written contiguously in each region (10 total copies of each configuration frame).

Configuration Frame Type	Frame Number
Header Frame	0x00000005 – 0x00000009 0x00000BAE – 0x00000BB2

Table 3-2 Configuration Frame Locations

3.2.2 Header Frame

The Header Frame contains information for identifying and navigating the tape. The data structure for the Header Frame is written in the data area of the frame.

Position	Bytes	Usage
0-7	8	Identification String
8	1	Major Revision
9	1	Minor Revision
10-15	6	reserved
16-275	260	Partition List
276-32767	32492	reserved

Figure 3-7 Header Frame Data Structure

3.2.2.1 Identification String

This field identifies the logical format by a null terminated ASCII text string. For this standard the entry shall be “ADR-SEQ”, indicating that it is the format compatible with the ADR implementation of the basic sequential access device.

3.2.2.2 Major Revision

Indicates the revision of this specification that was used in recording the tape. A change in major revision indicates that there may be compatibility problems or that older drives and drivers may be unable to take advantage of significant new features. All devices developed under this version of the specification (1.2) shall report a 1 in this field.

3.2.2.3 Minor Revision

Indicates the revision of this specification that was used in recording the tape. A change in minor revision indicates that there were minor or editorial changes made to the specification. There are no compatibility problems to be expected as a result of the change. All devices developed under this version of the specification (1.2) shall report a 2 in this field.

3.2.2.4 Partition List Structure

The Partition List structure resides in the Header Frame. A copy of the Partition Description for the current partition also exists in the AUX data of each frame.

Byte	Description
0	Number of Partitions (0x01)
1-3	reserved
4-259	**Partition Descriptions**

Figure 3-8 Partitions List

3.2.2.4.1 Number of Partitions

This field defines the total number of partitions on the tape. In the current revision of this specification, this field shall always be 0x01.

3.2.2.4.2 Partition Descriptions

The Partition Descriptions are copies of the Partition Description entries that are written in the AUX data of each frame with one additional field. A copy of the address for the last known EOD location is stored in the last entry.

Byte	Description
0	Partition Number
1	Partition Description Version (0x01)
2-3	Write Pass Counter
4-7	First Frame Address
8-11	Last Frame Address
12-15	EOD Frame Address

4. Command Set

4.1 Commands Supported

The DI-30, DP-30, SC-30 and SC-50 drives all use the same command set. The DI-30 and DP-30 also support the mandatory ATA commands. The drives do not support overlapped commands.

Table 4-1 Supported Packet Commands

Command		Code
Media Access	READ	08h
	WRITE	0Ah
	LOCATE	2Bh
	REWIND	01h
	LOAD / UNLOAD	1Bh
	WRITE ZERO FILEMARK	10h
Non Media Access	TEST UNIT READY	00h
	MODE SENSE	1Ah
	MODE SELECT	15h
	REQUEST SENSE	03h
	READ POSITION	34h
	INQUIRY	12h
	READ BUFFER	3Ch
	WRITE BUFFER	3Bh
	PREVENT / ALLOW MEDIA REMOVAL	E1h

Table 4-2 ATA commands supported

Command	Code	Error Register						Status Register				
		BBK	UNC	IDNF	ABRT	TK0NF	AMNF	DRDY	DWF	DSC	CORR	ERR
ATAPI Identify Device	A1				V			V		V		V
ATAPI Soft Reset	08											
ATAPI PktCommand	A0	Contains Packet Command Status						V			V	V
Set Features	EF				V			V	V	V		V
Execute Drive diagnostics	90	Special Drive Diagnostic Errors						V				V
NOP					V			V	V	V		V

V = valid on this command

4.2 Media Access Commands

4.2.1 Read

The Read command may be used to read information from the inserted media. A Read 0 command will put the device into the reading mode. The device will stop reading when the internal buffer is full, when the end of media is encountered, or when the device detects that there is no data recorded for 32 consecutive frames.

Bit Byte	7	6	5	4	3	2	1	0
0	Operation code (08h)							
1	Reserved						SILI Reserved	Fixed (1)
2	(MSB)							
3	Transfer Length							
4								(LSB)
5	Reserved							

Fixed : The drive supports fixed length blocks.

The Transfer Length specifies the number of Blocks. Blocks are either 32Kb or 32.5Kb in length (depending on selected block size). Transfer Length should be set to either 0x00 or 0x01 or else an INVALID FIELD IN CDB error is returned to the Host.

If a Read command is aborted and an UNRECOVERED READ ERROR is send to the Host. No data transfer phase has taken place. After the read-error the Host can continue reading. The Host can continue reading until a READ command is aborted with either an END OF PARTITION/MEDIUM DETECTED error or an END OF DATA DETECTED error.

On an END OF DATA DETECTED error the Device is unable to continue reading. The Host must issue a LOCATE command and a READ command to resume reading.

When no data is available in the buffer a Read command may hang up the IDE bus. It is recommended to first poll data availability before issuing a Read command. The Device can be forced to go into the read-mode with the Read 0 blocks command. No data phase and no IDE bus hang up will follow. Data availability can be checked either by issuing a MODE SENSE command on the buffer filling page or with a READ_POSITION command.

When the drive performed retries on a data block but successfully read the data, a message RECOVERED DATA WITH RETRIES is send to the Host after the data phase. The REQUEST SENSE Command Specific Information field contains the logical and physical position of the block and also the number of retries needed.
When the device is busy performing an previously requested operation that takes a long time the device will abort the READ command with a NOT READY IN PROGRESS BECOMING READY error message. Examples of these operations are an auto-load of a new inserted medium or a LOAD / UNLOAD / REWIND operation in progress. If the device is busy writing it's buffered data the device will abort the READ command with the NOT READY IN PROGRESS OF BECOMING READY error message.

When the device is busy flushing it's data buffers due to WRITE ZERO FILEMARKS command the device will abort the READ command with a NOT READY LONG WRITE error message.

Table 4-3 Error Codes for Read

Sense Key	ASC	ASCQ	Description of Error
00	00	02	END OF PARTITION/MEDIUM DETECTED
08	00	05	END-OF-DATA DETECTED
01	17	01	RECOVERED DATA WITH RETRIES
02	04	00	NOT READY, CAUSE NOT REPORTABLE
02	04	01	NOT READY IN PROGRESS BECOMING READY
02	04	02	NOT READY INITIALIZING COMMAND REQUIRED
02	04	08	NOT READY LONG WRITE
02	3A	00	MEDIUM NOT PRESENT
03	11	00	UNRECOVERED READ ERROR
05	24	00	INVALID FIELD IN CDB
06	28	00	NOT READY TO READY TRANSITION
06	29	00	POWER ON RESET OR DEVICE RESET OCCURRED

4.2.2 Write

The Write command may be used to write data to the inserted media. The device can accept write-data when the device is still busy locating to the requested position.

Bit Byte	7	6	5	4	3	2	1	0
0	Operation code (0Ah)							
1	Reserved							Fixed
2	(MSB)							
3	Transfer Length							
4								(LSB)
5	Reserved							

Fixed : The drive supports fixed length blocks.

The Transfer Length specifies the number of Blocks. Blocks are either 32Kb or 32.5Kb in length (depending on selected block size). Transfer Length should be set to either 0x00 or 0x01 or else an INVALID FIELD IN CDB error is send to the Host.

When the device is writing the Host must supervise the write process. During the write process two problems can occur. The device detects a medium flaw where it will report a write-failure. Or the device detects the end of media where it will report an medium-full error message.

The Host must poll for write completion by either reading out the buffer filling page with the MODE SENSE command or checking the buffer filling with the READ POSITION command. If the device reports zero blocks in buffer all data has been successfully written to tape. Note that the MODE SENSE command does not report write errors. Use WRITE 0 commands when the Host has finished writing but is waiting for write completion. The READ POSITION command does report write-errors.

It is recommended that the Host uses the reported buffer filling to prevent stalls on the IDE interface. When the buffer of the device is full a subsequent WRITE command may hang up the IDE bus. The host should only issue a write command when the device has buffer-space available. The Host can check for buffer availability by means of the reported buffer filling.

When the device reports a deferred MEDIUM ERROR: WRITE ERROR the device hasn't succeeded writing the buffered data to tape. When the write-error has been reported to the Host, the device has stopped writing and further error recovery must be done by the Host. The Request Sense Standard Data shows the physical and the logical position of the block where the device detected the write-error. The

number of blocks available in the buffer can either be requested with a MODE SENSE command on the buffer filling page or with the READ POSITION command. The data blocks in the buffer after the write error can be retrieved one by one with a READ BUFFER commands. The buffer contents must be discarded with either a MODE SELECT command on the buffer filling page or a LOCATE command. When the buffer is either read empty or cleared, the device is ready to start writing again.

In the Request Sense Standard Data of the write-error the device also reported a variable that informed the Host the number of frames to skip. It is recommended to skip this number of blocks before the device starts writing again. The new write position can be calculated by adding the 'Number Frames to Skip' to the write-error position.

When a MEDIUM ERROR: WRITE ERROR occurred on a WRITE 0x01 command no data has been transferred to the drive's buffer.
Writing can go on until a VOLUME OVERFLOW: END-OF-PARTITION/MEDIUM DETECTED error occurs. The WRITE command that returned the VOLUME OVERFLOW: END-OF-PARTITION/MEDIUM DETECTED error is not executed.

The Host must identify himself to the drive before the Host is able to write data to inserted medium. This must be done by means of the MODE SELECT command to the Vendor ID page (0x36). If not the device will abort the WRITE command with the error NOT READY INITIALIZING COMMAND REQUIRED.

When the device is busy performing an previously requested operation that takes a long time the device will abort the WRITE command with a NOT READY IN PROGRESS BECOMING READY error message. Examples of these operations are an auto-load of a new inserted medium or a LOAD / UNLOAD / REWIND operation in progress.

Table 4-4 Error Codes for Write

Sense Key	ASC	ASCQ	Description of Error
02	04	00	NOT READY, CAUSE NOT REPORTABLE
02	04	01	NOT READY IN PROGRESS BECOMING READY
02	04	02	NOT READY INITIALIZING COMMAND REQUIRED
02	3A	00	MEDIUM NOT PRESENT
03	0C	00	MEDIUM ERROR: WRITE ERROR
05	24	00	INVALID FIELD IN CDB
06	28	00	NOT READY TO READY TRANSITION
06	29	00	POWER ON RESET OR DEVICE RESET OCCURRED
07	27	00	WRITE PROTECTED
0D	00	02	VOLUME OVERFLOW: END-OF-PARTITION/MEDIUM DETECTED

4.2.3 Locate

The Locate command causes the device to position to the specified block. Prior to performing the locate operation, the host must ensure that all buffered data have been transferred to the medium.

Bit Byte	7	6	5	4	3	2	1	0
0	Operation code (2Bh)							
1	Reserved					BT (0)	0	Immed
2	Reserved							
3	(MSB)							
4	Logical block position							
5								
6								(LSB)
7	Reserved							
8	Reserved							
9	Reserved							

BT: A BT bit must be zero.

Immed: If the Immediate bit is set locate completion can be checked with Test Unit Ready commands. Until the locate is completed Test Unit Ready will report NOT READY IN PROGRESS BECOMING READY. If the Immediate bit is cleared subsequent TEST UNIT READY commands will report GOOD status even when the device is busy locating.

The Locate command positions the medium to the requested frame position. The device does not support relative addressing.

A Locate command returns ready immediately after command validation regardless of the state of the immediate bit.

When the device is busy performing a previously requested operation that takes a long time the device will abort the Locate command with a NOT READY IN PROGRESS BECOMING READY error message. Examples of these operations are an auto-load of a new inserted medium or a Load / Unload / Rewind operation in progress. If the device is busy writing it's buffered data, the device will abort the Locate command with the NOT READY IN PROGRESS BECOMING READY error message.

Table 4-5 Error Codes for Locate

Sense Key	ASC	ASCQ	Description of Error
02	04	00	NOT READY, CAUSE NOT REPORTABLE
02	04	01	NOT READY IN PROGRESS BECOMING READY
02	04	02	NOT READY INITIALIZING COMMAND REQUIRED
02	04	08	NOT READY LONG WRITE
02	3A	00	MEDIUM NOT PRESENT
04	15	00	RANDOM POSITIONING ERROR
03	0C	00	MEDIUM ERROR: WRITE ERROR
04	16	00	LOCATE NOT WITHIN PARTITION
05	24	00	INVALID FIELD IN CDB
06	28	00	NOT READY TO READY TRANSITION
06	29	00	POWER ON RESET OR DEVICE RESET OCCURRED

4.2.4 Rewind

The Rewind command causes the device to position to the load point position zero. Prior to performing the rewind operation, the host must ensure that all buffered data have been transferred to the medium.

Bit Byte	7	6	5	4	3	2	1	0
0	Operation code (01h)							
1	Reserved							Immed
2	Reserved							
3	Reserved							
4	Reserved							
5	Reserved							

Immed: The Immediate bit must be set to one. If not the device shall abort the command with INVALID FIELD IN CDB.

Until the Rewind command is completed Test Unit Ready will return NOT READY IN PROGRESS OF BECOMING READY.

When the device is busy performing a previously requested operation that takes a long time, the device will abort the Rewind command with a NOT READY IN PROGRESS BECOMING READY error message. Examples of these operations are a mount of a new inserted medium or a Load/Unload operation in progress. If the device is busy writing it's buffered data the device will abort the Rewind command with the NOT READY IN PROGRESS OF BECOMING READY error message.

Table 4-6 Error Codes for Rewind

Sense Key	ASC	ASCQ	Description of Error
02	04	00	NOT READY, CAUSE NOT REPORTABLE
02	04	01	NOT READY IN PROGRESS BECOMING READY
02	04	02	NOT READY INITIALIZING COMMAND REQUIRED
02	04	08	NOT READY LONG WRITE
02	3A	00	MEDIUM NOT PRESENT
04	15	00	RANDOM POSITIONING ERROR
05	24	00	INVALID FIELD IN CDB
06	28	00	NOT READY TO READY TRANSITION
06	29	00	POWER ON RESET OR DEVICE RESET OCCURRED

4.2.5 Load/Unload

The Load/Unload command requests that the drive enables or disables further media access operations. This command can also be used to request a re-tension function. The Unload command has options to open the tray of the device to receive media or eject media. To prevent tape wear the Load command can be used to park the head to a safe position. Prior to performing the load or unload operation, the host must ensure that all buffered data have been transferred to the medium.

Bit Byte	7	6	5	4	3	2	1	0
0	Operation code (1Bh)							
1	Reserved							Immed
2	Reserved							
3	Reserved							
4	Reserved			res	res	LoEj	Re-Ten	Load
5	Reserved							

Immed: The Immediate bit must usually be set to one except when a non-immediate standby is requested. In all other cases this bit must be set, if not the Device shall abort the command with INVALED FIELD IN CDB.

The Load command with the 'Force tray in standby position' option set is executed immediately. This command will park the head. This means that the head is removed from the tape.

Table 4-7 Load/Unload Available Functions

LoEj	Re-Ten	Load	Operation
0	0	0	Rewind
0	0	1	Load
0	1	0	Retention
0	1	1	Retention then mount
1	0	0	Rewind, then open the tray and eject tape
1	1	0	Retention, then open the tray and eject tape

A **Re**-**Ten**sion (**Re-Ten**) bit of one indicates that the medium in the device shall be retensioned. A retensioning function consists of one end-to-end pass on the cartridge.
If the **Load** bit is set to one, the medium shall be loaded and positioned to the frame 0. If the **Load** bit is zero, the medium in the device shall be positioned for removal at the Parking Zone. Following successful completion of an unload operation, the Device shall return either NOT READY INITIALIZING COMMAND REQUIRED or MEDIUM NOT PRESENT for all subsequent medium-access commands until a new volume is mounted or a load operation is successfully completed.

Completion of the Load/Unload command can be verified with Test Unit Ready command. Until the Load/Unload command is completed Test Unit Ready will return NOT READY IN PROGRESS BECOMING READY.

When the device is busy performing a previously requested operation the device will abort the Load/Unload command with a NOT READY IN PROGRESS BECOMING READY error message. Examples of these operations are an auto-load of a new inserted medium or a Rewind operation in progress. If the device is busy writing it's buffered data the device will abort the Load/Unload command with the NOT READY IN PROGRESS OF BECOMING READY error message.

Table 4-8 Error Codes for Load/Unload

Sense Key	ASC	ASCQ	Description of Error
02	04	00	NOT READY, CAUSE NOT REPORTABLE
02	04	01	NOT READY IN PROGRESS BECOMING READY
02	04	02	NOT READY INITIALIZING COMMAND REQUIRED
02	04	08	NOT READY LONG WRITE
02	3A	00	MEDIUM NOT PRESENT
05	24	00	INVALID FIELD IN CDB
06	28	00	NOT READY TO READY TRANSITION
06	29	00	POWER ON RESET OR DEVICE RESET OCCURRED

4.3 Non Media Access ATAPI functionality

4.3.1 Test Unit Ready

The Test Unit Ready command provides a means to check if the device is ready. This is not a request for a self-test. If the Device would accept an appropriate medium-access command without returning “Check Condition” status, this command shall return a “Good” status. If the Device cannot become operational or is in a state such that Host action is required to make the unit ready, the Device shall return “Check Condition” status with a Sense Key of NOT READY.

Bit Byte	7	6	5	4	3	2	1	0
0	Operation code (00h)							
1	Reserved							
2	Reserved							
3	Reserved							
4	Reserved							
5	Reserved							

The Test Unit Ready command is useful in that it allows the Host to poll the Device until it is ready without the need to allocate space for returned data. It is especially useful to check cartridge status. If a prior command is running upon receipt of this command such as an Rewind or a Load, the device shall report check condition with sense key 2 (NOT READY) and ASC/ASQ=04h/01h (IN PROGRESS OF BECOMING READY).

If the device is busy writing its buffered data, the device shall report GOOD status on Test Unit Ready commands. Except when the host issued a Write Filemarks command. In this case Test Unit Ready shall report NOT READY LONG WRITE until the write process is finished. Note: Normally Test Unit Ready will never report write-errors except when the host has previously issued a Write Filemarks command. If the device has detected a write-error and the host has issued a Write Filemarks, the device will report a MEDIUM ERROR: WRITE ERROR on a Test Unit Ready command.

When the device is busy performing a auto-load of a new inserted medium the device will report NOT READY IN PROGRESS OF BECOMING READY on a Test Unit Ready command. When the device successfully auto-loaded the medium the device will report a unit attention with the additional sense code set to NOT READY TO READY TRANSITION.

Table 4-9 Error Codes for Test Unit Ready

Sense Key	ASC	ASCQ	Description of Error
00	00	00	NO ADDITIONAL SENSE INFORMATION
05	24	00	INVALID FIELD IN CDB
02	04	00	NOT READY – CAUSE NOT REPORTABLE
02	04	01	NOT READY - IN PROGRESS OF BECOMING READY
02	04	02	NOT READY – INITIALIZING COMMAND REQUIRED
02	3A	00	MEDIUM NOT PRESENT
02	04	08	NOT READY – LONG WRITE IN PROGRESS
03	0C	00	MEDIUM ERROR: WRITE ERROR (see note)
06	28	00	NOT READY TO READY TRANSITION
06	29	00	POWER ON RESET OR DEVICE RESET OCCURRED

4.3.2 Mode Sense

Mode Sense returns the requested mode page in order that configuration or ADR unique status may be determined.

Bit Byte	7	6	5	4	3	2	1	0
0	Operation code (1Ah)							
1	Reserved				DBD (1)	Reserved		
2	PC(00b)		Page Code					
3	(MSB) Allocation Length							
4	(LSB)							
5	Reserved							

DBD : The drive does not support Block Descriptors so this bit should be set TRUE.
Page Code : specifies which Page is returned
PC : The drive only supports current values. PC = 00b always
Allocation Length :

Number [0000h .. FFFFh] Specifies the maximum number of bytes that the Host has allocated for returned data.

Table 4-10 Error Codes for Mode Sense

Sense Key	ASC	ASCQ	Description of Error
02	04	00	NOT READY, CAUSE NOT REPORTABLE
05	24	00	INVALID FIELD IN CDB
06	28	00	NOT READY TO READY TRANSITION
06	29	00	POWER ON RESET OR DEVICE RESET OCCURRED

4.3.2.1 Mode Sense Data Format

Bit Byte	7	6	5	4	3	2	1	0
	Mode Parameter Header							
	Page							

4.3.2.2 Mode Page Header

Bit Byte	7	6	5	4	3	2	1	0
0	Mode Data Length							
1	Medium Type							
2	WP	1(Buffered Mode)			Reserved			
3	Block Descriptor Length (0)							

Mode Data Length : Number specifies the length in bytes of the data that follows (not including itself).

Medium Type : 0xDA

WP : Write Protect status of current medium

Block Descriptor Length : Number specifies length in bytes of the block descriptor, since their is no block descriptor it is set to 00h.

4.3.2.3 Mode Page Format

Bit Byte	7	6	5	4	3	2	1	0
0	PS	res	Page Code					
1	Page Length (n - 1)							
2 .. n	Mode Parameters							

Page Code :
Number [00h .. 34h] specifies format and parameters for that mode page
See Table 4-11
PS : Parameters Savable
H : Mode Page can be saved
L : supported parameters cannot be saved
Page Length :
Number [00h .. FFh] specifies the length in bytes of the Mode Parameters that follow

Table 4-11: Supported Mode Pages

Page Code	Description
2Ah	Capabilities and Mechanical Status page
2Bh	Tape Parameters page
2Fh	Number Retries page
30h	Data Transfer Mode page
33h	Buffer Filling page
36h	Vendor Identification page
37h	Locate Status Page

4.3.2.4 Capabilities and Mechanical Status Page

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved		Page Code (2Ah)					
1	Page Length (12h)							
2	Reserved							
3	Reserved							
4	Reserved	Reserved	SPREV (0)	Reserved	Reserved	Reserved	Reserved	RO
5	Reserved	Reserved	QFA (0)	Reserved	EFMT (0)	Reserved	Reserved	Reserved
6	CMPRS (0)	ECC (1)	Reserved	DISCON - NECT (0)	EJECT (1)	PREVENT (0)	LOCKED	LOCK (1)
7	BLK32768 (1)	Reserved	Reserved	Reserved	Reserved	BLK1024 (0)	BLK512 (0)	Reserved
8	(MSB) Maximum Speed Supported (in 1000 bytes/s)							
9								(LSB)
10	Reserved							
11								
12	(MSB) Continuous Transfer Limit (in blocks)							
13								(LSB)
14	(MSB) Current Speed Selected (in 1000 bytes/s)							
15								(LSB)
16	(MSB) Buffer Size (in 512 bytes)							
17								(LSB)
18	Reserved							
19								

The **SPREV** bit is not set, the Devices does not support SPACE in the reverse direction.
If the **RO** bit is set, the Device is operating in a read only mode. This bit does not reflect the state of the write protect mechanism of the cartridge which is indicated by the WP bit in the Mode Page Header.
The **QFA** bit is not set. The ADR accommodates new ways to store Partition information.
The **EFMT** bit is not set, the Device does not support ERASE command initiated formatting.
The **CMPRS** bit is not set, the Device does not support data compression.
The **ECC** bit is set, the Device performs error correction.
ATAPI Command Overlap is not supported so the **DISCONNECT** bit is set to zero.
The **EJECT** bit is set, the Device can mechanically unload the volume with the Load/Unload command.
The **PREVENT** bit is not set, the Device does not default in the Prevent state after power up.
If the **LOCKED** bit is set, the volume is locked.
The **LOCK** bit is set, the Device supports locking of a volume using the Prevent/Allow Medium Removal command.
The **BLK32768** bit is set, the Device is capable of using a 32768 byte block size.
The **BLK1024** bit is not set, the Device is not capable of using a 1024 byte block size.
The **BLK512** bit is not set, the Device is not capable of using a 512 byte block size.
The **Maximum Speed Supported** field indicates the maximum data rate the Device supports. This value is returned in 1000 bytes per second units which corresponds to the maximum sustained data transfer rate the Device is capable of without consideration of data compression. It is set to 0831h.
The **Continuous Transfer Limit** field indicates the number of blocks for the current block size that can be transferred without delay due to a buffer limitation. Transfers restricted to this size when **DISCONNECT** is set to zero will result in efficient use of the bus. This number is set to 0001h.
The **Current Speed Selected** field indicates the actual data rate that the Device is currently using this value is return as 1000 bytes per second that the data is transferred between the host and the drive.
The **Buffer Size** is an estimate in 512 byte increments of the read and write buffer size. The buffer size is set to 1000h

4.3.2.5 Tape Parameters Page (ADR unique)

Bit Byte	7	6	5	4	3	2	1	0
0	PS(0)	Reserved	Page Code (2Bh)					
1	0Eh							
2	Density (Kbpi)							
3	Reserved							
4	Reserved							
5	BSEG (1)							
6	(MSB) SEGTRK							
7								(LSB)
8	(MSB) TRKS							
9								(LSB)
10	Reserved							
11	Reserved							
12	Reserved							
13	Reserved							
14	Reserved							
15	Res	Res	Res	Res	Res	Res	Res	Res

This page is used to determine the capacity of the tape that has been inserted into the drive. The SEGTRK is the number of frames per track on the inserted Medium. The TRKS is the number of tracks the drive can address. This page will only be valid after the drive has mounted a medium successfully. The host can poll the state of the mounting process with the Test Unit Ready command. If the drive returns GOOD status on a Test Unit Ready command, this page can be read.

The total number of frames on tape is SEGTRK*TRKS. In the case of COT load point tapes (25GB and higher) total user frames available = (SEGTRK–99)*TRKS, which excludes the center of tape parking zone where no user data can be recorded.

4.3.2.6 Number of Retries Page (ADR unique)

Bit Byte	7	6	5	4	3	2	1	0
0	PS(1)	Reserved	Page Code (2Fh)					
1	2							
2						PER		
3	Max Number of Retries							

PER : A Post Error bit of one indicates that the device shall report recovered errors. A PER bit of zero indicates that the device shall not report recovered errors. A PER bit of one indicates the device shall report recovered errors. The default value is zero.

Max Number of Retries : This field will return the maximum number of retries the drive will perform when reading data. The default number of retries is 5.

The Max Number of Retries specifies the number of tape repositions when reading. When the Max Number of Retries is exhausted and the device failed to read the data correctly a read error is send to the host.

4.3.2.7 Data Transfer Mode Page (ADR unique)

Bit Byte	7	6	5	4	3	2	1	0
0	PS(1)	Reserved	Page Code (30h)					
1	2							
2	Reserved							
3	Streaming Mode (1)	Reserved	32.5Kb Record (1)	32Kb Record (0)	Reserved		32.5Kb Playback (1)	32Kb Playback (0)

Streaming Mode: This bit enables the drive to automatically turn tracks at the proper boundaries when streaming, for the purpose of this use model this bit must always be set to one (1).
32Kb Record: Only the frame's user data is transferred from the host to the drive. For the purpose of this use model this bit must always be cleared to zero (0).
32.5Kb Record: Data followed by AUX is transferred from the host to the drive. For the purpose of this use model this bit must always be set to one (1).
32Kb Playback: Only the frame's user data is transferred from the drive to the host. For the purpose of this use model this bit must always be cleared to zero (0).
32.5Kb Playback: Data followed by AUX is transferred from the drive to the host. For the purpose of this use model this bit must always be set to one (1).

4.3.2.8 Buffer Filling Page (ADR unique)

Bit Byte	7	6	5	4	3	2	1	0
0	PS(0)	Reserved	Page Code (33h)					
1	2							
2	Maximum Number of Frames in Buffer							
3	Current Number of Frames in Buffer							

Maximum Number of Frames in Buffer :
Tells the host the maximum number of frames the Main buffer can contain when reading or writing.
Current Number of Frames in Buffer :
Tells the Host the number of frames currently in the buffer at time of issuing the command.

The command shall return "Check Condition" status with a Sense Key of MEDIUM NOT PRESENT when no volume is mounted, or NOT READY IN PROGRESS OF BECOMMING READY when the Device is busy mounting a volume.

4.3.2.9 Vendor Identification Page (ADR unique)

Bit Byte	7	6	5	4	3	2	1	0
0	PS(1)	Reserved	Page Code (36h)					
1	6							
2	**(MSB)** Vendor Identification String							
3								
4								
5								(LSB)
6	Reserved							
7	Reserved							

The Vendor Identification String is used to provide a means for the application to identify tapes.

By means of this page the host can retrieve the Vendor Identification String it has programmed with the Mode Select command. When the device is writing the Vendor Identification String will be embedded in the AUX field of the write data and can be retrieved from the media when the drive is reading.

4.3.3 Mode Select

This command is used by the host to change the default settings or to initialize parameters of the drive.

Bit Byte	7	6	5	4	3	2	1	0
0	Operation code (15h)							
1	Reserved			PF(1)	Reserved			SP(0)
2	Reserved							
3	(MSB) Parameter List Length							
4								(LSB)
5	Reserved							

SP : The Device is not capable to save pages to non-volatile memory
PF : Always set to one
Parameter List Length :
Number [0000h .. FFFFh] specifies the length in bytes of the data that shall be transferred to the drive.

Table 4-12 Error Codes for Mode Select

Sense Key	ASC	ASCQ	Description of Error
02	04	00	NOT READY, CAUSE NOT REPORTABLE
05	1A	00	PARAMETER LIST LENGTH ERROR
05	24	00	INVALID FIELD IN CDB
05	26	00	INVALID FIELD IN PARAMETER LIST
05	26	01	PARAMETER NOT SUPPORTED
05	26	02	PARAMETER VALUE INVALID
06	28	00	NOT READY TO READY TRANSITION
06	29	00	POWER ON RESET OR DEVICE RESET OCCURRED

4.3.3.1 Mode Select Data Format

Bit Byte	7	6	5	4	3	2	1	0
	Mode Parameter Header							
	Page							

4.3.3.2 Mode Parameter Header

Bit Byte	7	6	5	4	3	2	1	0
0	Mode Data Length							
1	Medium Type							
2	Reserved							
3	Block Descriptor Length(00h)							

Mode Data Length : Number [00h .. FFh] specifies the length in bytes of data that is available to be transferred Note the Mode Data Length does not include itself
Medium Type : Specified in QIC 95-101
Device Specific Parameter : For MODE SELECT this field is ignored i.e. Reserved
Block Descriptor Length : 00h : No Block Descriptors will be transferred on a MODE SELECT command

4.3.3.3 Mode Page Format

Bit Byte	7	6	5	4	3	2	1	0
0	PS	Reserved	Page Code					
1	Page Length (n - 1)							
2 .. n	Mode Parameters							

Page Code: specifies the page
Page Length: specifies the page parameter length in bytes

Table 4-13: Supported Mode Pages for Mode Select

Page Code	Description
2Fh	Number Retries page
30h	Data Transfer Mode page
33h	Buffer Filling page
36h	Vendor Identification page

4.3.3.4 Number of Retries Page (ADR unique)

Bit Byte	7	6	5	4	3	2	1	0
0	PS(1)	Reserved	Page Code (2Fh)					
1	2							
2						PER		
3	Max Number of Retries							

A Post Error (PER) bit of one indicates that the device shall report recovered errors. A PER bit of zero indicates that the device shall not report recovered errors. A PER bit of one indicates the device shall report recovered errors. The default value is zero.

The Max Number of Retries specifies the number of tape repositions when reading data from the medium. When the number of retries are
exhausted a read error is send to the Host.

4.3.3.5 Data Transfer Mode Page (ADR unique)

Bit Byte	7	6	5	4	3	2	1	0
0	PS(1)	Reserved	Page Code (30h)					
1	2							
2	Reserved							
3	Streaming Mode (1)	Reserved	32.5Kb Record (1)	32Kb Record (0)	Reserved		32.5Kb Playback (1)	32Kb Playback (0)

Streaming Mode: This bit enables the drive to automatically turn tracks at the proper boundaries when streaming, for the purpose of this use model this bit must always be set to one (1).
32Kb Record: Only the frame's user data is transferred from the host to the drive. For the purpose of this use model this bit must always be cleared to zero (0).
32.5Kb Record: Data followed by AUX is transferred from the host to the drive. For the purpose of this use model this bit must always be set to one (1).
32Kb Playback: Only the frame's user data is transferred from the drive to the host. For the purpose of this use model this bit must always be cleared to zero (0).
32.5Kb Playback: Data followed by AUX is transferred from the drive to the host. For the purpose of this use model this bit must always be set to one (1).

4.3.3.6 Buffer Filling Page (ADR unique)

Bit Byte	7	6	5	4	3	2	1	0
0	PS(1)	Reserved	Page Code (33h)					
1	2							
2	Reserved							
3	Number Frames to clear from Buffer							

This page can be used to clear the write data available in the drive's data buffer (delete all data queued data from the buffer). Special use for this page is meant for a MEDIUM ERROR: WRITE_ERROR. If the Number Frames to clear from Buffer is higher than the actual number of frames in the buffer the command is aborted and a INVALID FIELD IN PARAMETER LIST error is returned. The available number of frames in the buffer can be retrieved with the MODE SENSE command on the Buffer Filling page.

4.3.3.7 Vendor Identification Page (ADR unique)

Bit Byte	7	6	5	4	3	2	1	0
0	PS(1)	Res	Page Code (36h)					
1	6							
2	(MSB)							
3	Vendor Identification string							
4								
5								(LSB)
6								
7								

By means of this page the host must identify itself to the drive. The drive will not perform any media access command until the host has sent his 4 byte identify string.

The Vendor Identification String is used to provide means for the host Software to identify his Tapes. The device will ensure that in every block written, the connected AUX field of that block, will have an copy of the Vendor Identification string. See ADR Physical Tape Format for more information.

The Host can retrieve the Vendor Identification string it has programmed with the Mode Sense command. The Vendor Identification String is embedded in the write data and can be retrieved from the media when the drive is reading.

4.3.4 Read Position

The Read Position command reports the current position of the drive. No medium movement shall occur as a result of the command.

Bit Byte	7	6	5	4	3	2	1	0
0	Operation code (34h)							
1	Reserved							BT (0)
2	Reserved							
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7	Reserved							
8	Reserved							
9	Reserved							

4.3.4.1 Read Position Data Returned

Bit Byte	7	6	5	4	3	2	1	0
0	BOP	EOP	Reserved					
1	Reserved							
2	Reserved							
3	Reserved							
4	(MSB)							
5	First Frame Position							
6								(LSB)
7								
8	(MSB)							
9	Last Frame Position							
10								(LSB)
11								
12	Reserved							
13	Reserved							
14	Reserved							
15	Blocks in Buffer							
16-19	Reserved							

A **B**eginning **O**f **P**artition (**BOP**) bit of one indicates that the drive is at the first frame on the tape.
An **E**nd **O**f **P**artition (**EOP**) bit of one indicates that the drive is at the last frame on the tape.

The First Frame Position is the address of the data that will be transferred to or from the host.

The Last Frame Position is the address of the data that will be transferred to or from the tape.

When the drive is in write mode the Blocks in Buffer represent the number of blocks not yet written to tape. When in the drive is in read mode Blocks in Buffer represents the number of blocks read from tape but not sent to the host.

Table 4-14 Error Codes for Read Position

Sense Key	ASC	ASCQ	Description of Error
02	04	00	NOT READY, CAUSE NOT REPORTABLE
02	3A	00	MEDIUM NOT PRESENT
03	0C	00	MEDIUM ERROR: WRITE ERROR
05	24	00	INVALID FIELD IN CDB
06	28	00	NOT READY TO READY TRANSITION
06	29	00	POWER ON RESET OR DEVICE RESET OCCURRED
0D	00	02	VOLUME OVERFLOW: END-OF-PARTITION/MEDIUM DETECTED

4.3.5 Request Sense

The Request Sense command requests that the drive transfer sense data to the host.

Bit Byte	7	6	5	4	3	2	1	0
0	Operation code (03h)							
1	Reserved							
2	Reserved							
3	Reserved							
4	Allocation Length							
5	Reserved							

Allocation Length: The length in bytes that the host is able to accommodate.

The sense data shall be available if an error condition ("Check Condition") had previously been reported to the host. If the drive has no other sense data available to return, it shall return a Sense Key of NO SENSE and an Additional Sense Code of NO ADDITIONAL SENSE INFORMATION.

The sense data shall be preserved by the drive until retrieved by a Request Sense command or until the receipt of any other I/O Command. The device returns 16 bytes of data in response to a REQUEST SENSE command. If the allocation length is 20 or greater, the host should assume that the bytes not transferred would have been zeros had the drive returned those bytes.

Table 4-15 Error Codes for Request Sense

Sense Key	ASC	ASCQ	Description of Error
05	24	00	INVALID FIELD IN CDB
06	28	00	NOT READY TO READY TRANSITION
06	29	00	POWER ON, RESET OR DEVICE RESET OCCURRED

4.3.5.1 Request Data Returned

Bit Byte	7	6	5	4	3	2	1	0
0	Valid (0)	Error Code (70h or 71h)						
1	Reserved (Segment Number)							
2	Reserved	EOM	ILI	Reserved	Sense Key			
3	*(MSB)*							
4	Information							
5	*(Frame Position of encountered error)*							
6	*(LSB)*							
7	Additional Sense Length (8)							
8	*(MSB)*							
9	Command Specific Information							
10								
11	*(LSB)*							
12	Additional Sense Code							
13	Additional Sense Code Qualifier							
14	Field Replaceable Unit Code (00h)							
15	SKSV(0)	Sense Key Specific (00h)						

Error Code :
Number [70h, 71h]

70h : current error
71h : deferred error
Valid :
H : The **information** field contains valid information as specified in the QIC 157 Rev D standard
L : The **information** field contain information different from the information specified in the QIC157 Rev D standard
Sense Key :
Number [00h .. 0Fh] indicates generic information describing an error or exception
ILI : Incorrect Length Indicator
H : requested allocation length or selected block size did not match logical block length of the data on the medium
L : no length error occurred
EOM : End Of Medium
H : end of medium condition exists
L : no end of medium condition
Additional Sense Length :
Number 08h indicates the number of additional sense bytes to follow always 8 bytes long
Additional Sense Code :
Number [00h .. FFh] indicates further information related to the error or exception reported by the Sense Key (See *"QIC 157 Rev D standard"*)

Information :
This field is valid when a READ or WRITE command completed with a check condition. When valid the Information field specifies the Frame Position of the encountered error. The possible check conditions on which this field contains valid data are :

RECOVERED DATA WITH RETRIES
UNRECOVERED READ ERROR
MEDIUM ERROR: WRITE ERROR

Command Specific Information :
This field is valid when a READ or WRITE command completes with a check condition. The possible check conditions are described above in the information field. When valid the Command Specific and in the case of a MEDIUM ERROR: WRITE ERROR this field contains the advised number of blocks to skip. In case of a read error this field contains the number of retriesattempted. See the next two tables for more information.

4.3.5.2 Command Specific Information on MEDIUM ERROR: WRITE ERROR

Byte	Description
8	Reserved
9	Advised number of blocks to skip
10-11	Reserved

4.3.5.3 Command Specific Information on UNRECOVERED READ ERROR or RECOVERED DATA WITH RETRIES

Byte	Description
8	Reserved
9	Performed number of retries
10-11	Reserved

Additional Sense Code Qualifier :
Number [00h .. FFh] indicates detailed information related to the Additional Sense Code (See *"QIC 157 Rev D standard"*)
Field Replaceable Unit Code :
Number [00H .. FFH] specifies a Device specific mechanism or unit that has failed
00H : indicates that no specific mechanism or unit has been identified to have failed
SKSV : Sense Key Specific Valid
L : always
Sense Key Specific :
Not used .

4.3.6 Inquiry

Inquiry provides basic device identification.

Bit Byte	7	6	5	4	3	2	1	0
0	Operation code (12h)							
1	Reserved							Reserved (EVPD)
2	Reserved (Page Code)							
3	Reserved							
4	Allocation Length							
5	Reserved							

Allocation Length :
Number [00h .. FFh] indicates the length in bytes that the Host is able to accommodate

4.3.6.1 Inquiry Data Returned

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved (Peripheral Qualifier)			Peripheral Device Type 01h				
1	RMB 1	Reserved						
2	ISO Version (*tbd)*		ECMA Version (*tbd*)			ANSI Version (02h)		
3	Reserved (AENC)	Reserved (TrmIOP)	Reserved		Response Data Format (02h)			
4	Additional Length (47-4)							
5	Reserved							
6	Reserved							
7	Reserved	WBus32 0	Wbus16 1	Sync 0	Reserved	Reserved	CmdQue 0	SftReset 1
8-15	Vendor Identification							
16-31	Product Identification							
32-35	Revision							

Peripheral Device Type :
Number [00h .. 1Fh] indicates the type of device
RMB : Removable Medium Bit
H : removable
L : not removable
ANSI Version :
Number [0h .. 7h]
ECMA Version :

Number [0h .. 7h]
ISO Version :
Number [0h .. 3h]
Response Data Format :
Number [00h .. 0Fh]
Additional Length :
Number: specifies the number in bytes of parameters to follow
Sync :
H : device supports synchronous data transfers
L : device does not support synchronous data transfers
WBus16 :
H : device supports 16 bit wide transports
L : device does not support 16 bit wide transports
WBus32:
H : device supports 32 bit wide transports
L : device does not support 32 bit wide transports
Vendor Identification:
8 bytes char string
Product Identification:
16 bytes char string
Revision:
4 bytes char string

Table 4-16 Error Codes for Inquiry

Sense Key	ASC	ASCQ	Description of Error
05	24	00	INVALID FIELD IN CDB

4.3.7 Write Filemark

The ADR DI, DP, and SC drives do not support writing filemarks. The Write Filemark command is only used to flush the write buffer to tape.

Bit Byte	7	6	5	4	3	2	1	0
0	Operation code (10h)							
1	Reserved							Immed
2	Reserved							
3	Reserved							
4	Reserved							
5	Reserved							

The immed bit must be set. If not the drive shall return “Check Condition” status with the Sense key set to INVALID FIELD IN CDB.

This command is used to force flushing of buffered write data. When the drive is in the process of flushing buffers Test Unit Ready can be used to poll when the flush is finished. Test Unit Ready will respond with NOT READY – LONG WRITE IN PROCESS until the buffer is flushed.

Table 4-17 Error Codes for Write Filemark

Sense Key	ASC	ASCQ	Description of Error
02	04	00	NOT READY, CAUSE NOT REPORTABLE
02	3A	00	MEDIUM NOT PRESENT
03	0C	00	MEDIUM ERROR: WRITE ERROR
05	24	00	INVALID FIELD IN CDB
06	28	00	NOT READY TO READY TRANSITION
06	29	00	POWER ON RESET OR DEVICE RESET OCCURRED
07	27	00	WRITE PROTECTED
0D	00	02	VOLUME OVERFLOW: END-OF-PARTITION/MEDIUM DETECTED

4.3.8 Prevent/Allow Medium Removal

By means of this command the host can either enable or disable the removal of medium from the ADR.

Byte\Bit	7	6	5	4	3	2	1	0
0	Operation Code (1Eh)							
1	Reserved							
2	Reserved							
3	Reserved							
4	Reserved							Prevent
5	Reserved							

Prevent :
H : medium removal prevented
L : medium removal allowed

The ADR is equipped with a Loader to provide a locking mechanism for an inserted cartridge. While a prevention of medium removal is in effect, the drive inhibits removal of the medium by an operator using the front panel button.

The default state of the drive at power on is unlocked.

Table 4-18 Prevent/Allow Medium Removal State Changes

Operation	Locked / Unlocked	If Drive Not Ready (No Media)	If Drive Ready (Media Present)
Unlock	Unlocked	No error	No error
(Prevent=0)	Locked	No error, now media may be inserted	No error, now media may be removed
Lock	Unlocked	No error, drive door locked and will not allow media to be inserted	No error, drive door locked and will not allow media to be inserted
(Prevent=1)	Locked	No error	No error
UNLOAD with	Unlocked	No error and tray is opened	No error: media ejects
LoEj = 1	Locked	Error : MEDIA REMOVAL PREVENTED	Error : MEDIA REMOVAL PREVENTED
Manual Eject	Unlocked	Tray opens	Media is ejected
	Locked	Deferred error : MEDIA REMOVAL PREVENTED. No operation	Deferred error : MEDIA REMOVAL PREVENTED. No operation
Manual Load	Unlocked	No operation	Pseudo LOAD

Table 4-19 Error Codes for Prevent/Allow Medium Removal

Sense Key	ASC	ASCQ	Description of Error
05h	24h		INVALID FIELD IN CDB
06h	28h		NOT READY TO READY TRANSITION
06h	29h		POWER ON, RESET OR BUS DEVICE RESET OCCURRED
02h	04h	00h	NOT READY - CAUSE IS NOT REPORTABLE
02h	04h	01h	NOT READY - IN PROGRESS OF BECOMING READY
02h	04h	03h	LOGICAL DRIVE NOT READY - MANUAL INTERVENTION REQUIRED
02h	3Ah		MEDIUM NOT PRESENT
02h	53h	02h	MEDIA REMOVAL PREVENTED

4.4 Summary of Error Codes

Table 4-20: Sense Key, ASC and ASCQ for errors

Sense Key	ASC	ASCQ	Description of Error
00	00	00	NO ADDITIONAL SENSE INFORMATION
00	00	02	END OF PARTITION/MEDIUM DETECTED
01	17	01	RECOVERED DATA WITH RETRIES
02	04	00	NOT READY, CAUSE NOT REPORTABLE
02	04	01	NOT READY - IN PROGRESS OF BECOMING READY
02	04	02	NOT READY – INITIALIZING COMMAND REQUIRED
02	04	03	NOT READY – MANUAL INTERVENTION REQUIRED
02	04	04	NOT READY - NO EMBOSSED HEADER
02	04	08	NOT READY – LONG WRITE
02	3A	00	MEDIUM NOT PRESENT
02	53	02	MEDIA REMOVAL PREVENTED
03	0C	00	MEDIUM ERROR: WRITE ERROR
03	11	00	UNRECOVERED READ ERROR
04	15	00	RANDOM POSITIONING ERROR
04	16	00	LOCATE NOT WITHIN PARTITON
04	47	00	SCSI BUS PARITY ERROR
05	1A	00	PARAMETER LIST LENGTH ERROR
05	20	00	COMMAND NOT IMPLEMENTED
05	24	00	INVALID FIELD IN CDB
05	25	00	ILLEGAL LUN
05	26	00	INVALID FIELD IN PARAMETER LIST
05	26	01	PARAMETER NOT SUPPORTED
05	26	02	PARAMETER VALUE INVALID
05	3D	00	ILLEGAL ID MESSAGE
06	28	00	NOT READY TO READY TRANSITION
06	29	00	POWER ON RESET OR DEVICE RESET OCCURRED
07	27	00	WRITE PROTECTED
08	00	05	END-OF-DATA DETECTED
0D	00	02	VOLUME OVERFLOW: END-OF-PARTITION/MEDIUM DETECTED
04	40	01	DIAGNOSTICS FAILURE- CALIBRATION ERROR
04	40	02	DIAGNOSTICS FAILURE- WRITE DUMMY FRAME
04	40	03	DIAGNOSTICS FAILURE- HEADER READING ERROR
04	40	04	DIAGNOSTICS FAILURE- WPFAIL DUMMYFRAME
04	40	05	DIAGNOSTICS FAILURE- LOCK DUMMYFRAME
04	40	06	DIAGNOSTICS FAILURE- COMMAND FIFO FULL
04	40	07	DIAGNOSTICS FAILURE- BUFFER PROTECT ERROR
04	40	08	DIAGNOSTICS FAILURE- WRITE IC FAILED
04	40	09	DIAGNOSTICS FAILURE- FEDONE TIMEOUT
04	40	0A	DIAGNOSTICS FAILURE- READ IC FAILED
04	40	0B	DIAGNOSTICS FAILURE- L SENSE REGULATION
04	40	0C	DIAGNOSTICS FAILURE- SERVO IDENTIFY TIMEOUT
04	40	0D	DIAGNOSTICS FAILURE- CMD-CTRL TIMEOUT
04	40	0E	DIAGNOSTICS FAILURE- FACTORY DEFAULTS DETECTED

4.5 ATA Commands

4.5.1 ATAPI Packet Command

COMMAND CODE - A0h

4.5.2 ATAPI Identify Device

COMMAND CODE - A1h

PROTOCOL - PIO data in.

INPUTS -

Register	7	6	5	4	3	2	1	0
Features								
Sector Count								
Sector Number								
Cylinder Low								
Cylinder High								
Device/Head	1		1	D				
Command	A1h							

NORMAL OUTPUTS - None

ERROR OUTPUTS - None.

Status register				Error register					
DRDY	DF	CORR	ERR	BBK	UNC	IDNF	ABRT	TK0NF	AMNF
V									

PREREQUISITES - DRDY set to one.

DESCRIPTION - The ATAPI IDENTIFY DRIVE command enables the Host to receive parameter information from the ADR.

The main purpose for this command is not to determine the medium capacity but to initialize the IDE PIO and the DMA mode for the Host Chip set. Note information regarding medium capacity should be requested with the MODE SENSE Command

4.5.2.1 ATAPI Identify Drive parameter information

Word		Description
0		General configuration bit significant 0x0080
1		Number of logical cylinders
2		
3		Number of logical heads
4		Reserved
5		
6		Number of logical sectors per logical track
7-9		Vendor Specific

10-19		Serial Number
20		Vendor Specific
21		Vendor Specific
22		
23-26		Firmware revision (8 ASCII characters)
27-46		Model Number (40 ASCII characters)
47		Multiple Sector Command, Sector Count
48		Reserved
49		Capabilities 15 -14 Reserved 13 0 = Standby timer values are vendor specific 12 Reserved 11 1 = IORDY supported 10 0 = IORDY cannot be disabled 9 0 = No Logical Block Addressing 8 1 = DMA supported 7 - 0 Vendor specific
50		Reserved
51		15 - 8 0x04 = PIO data transfer cycle timing mode 7 - 0 Vendor specific
52		15 - 8 0x02 = DMA data transfer cycle timing mode 7 - 0 Vendor specific
53		15 - 2 Reserved 1 0 = fields reported in words 64-70 are not valid 0 0 = fields reported in words 54-58 are not valid
54		Number of current logical cylinders
55		Number of current logical heads
56		Number of current logical sectors per track
57-58		Current capacity in sectors
59		Reserved
60-61		Total number of User Addressable Sectors
62		15 - 11 Reserved 10 - 8 Indication of what DMA mode is selected 7 - 3 Reserved 2 - 0 0x7 Indication of the possible DMA modes
63		15 - 12 Reserved 11 - 8 Indication of what PIO mode is selected 7 - 4 Reserved 3 - 0 0xF = Indication of the possible PIO modes
64		15 - 8 Reserved 7 - 0 Advanced PIO Transfer Mode Supported
65		Minimum Multiword DMA Transfer Cycle Time Per Word (ns)
66		Manufacturer's Recommended Multiword DMA Transfer Cycle Time (ns)
67		Minimum PIO Transfer Cycle Time without Flow Control
68		Minimum PIO Transfer Cycle Time with IORDY Flow Control
69-70		Reserved
71-72		Reserved
73		Major Revision Number
74		Minor revision Number
75-128		Reserved
128-159		Vendor Specific
160-255		Reserved

4.5.3 ATAPI Soft Reset

COMMAND CODE - 08h

PROTOCOL - Non-data

INPUTS -

Register	7	6	5	4	3	2	1	0
Features								
Sector Count								
Sector Number								
Cylinder Low								
Cylinder High								
Device/Head	1		1	D				
Command	08h							

NORMAL OUTPUTS -

Register	7	6	5	4	3	2	1	0
Error	01h							
Sector Count	01h							
Sector Number	01h							
Cylinder Low	14h							
Cylinder High	EBh							
Device/Head	0	0	0	D	0	0	0	0
Status	00h							

ERROR OUTPUTS - None.

Status register				Error register					
DRDY	DF	CORR	ERR	BBK	UNC	IDNF	ABRT	TK0NF	AMNF

PREREQUISITES - None.

DESCRIPTION - The SOFT RESET PACKET command enables the host to reset an individual device without affecting the other device.

Upon receipt of this command, the device shall set BSY to one, and perform the reset sequence described in Paragraph 9.2 in the ATA2 Revision 3 Standard, except for the PDIAG- handshake. When completed, the device shall clear BSY to zero and the register contents shall be as shown in NORMAL OUTPUTS.

4.5.4 Set Features

COMMAND CODE - EFh

PROTOCOL - Non-data.

INPUTS - The subcommand "03h : Set transfer mode based on value in Sector Count register.." Is allowed. Table 80 defines the possible subcommand specific values

Register	7	6	5	4	3	2	1	0
Features	03							
Sector Count	Subcommand specific							
Sector Number								
Cylinder Low								
Cylinder High								
Device/Head	1		1	D				
Command	EFh							

NORMAL OUTPUTS - See the subcommand descriptions.

ERROR OUTPUTS - If any subcommand input value is not supported or is invalid, the device posts an Aborted Command error.

Status register				Error register					
DRDY	DF	CORR	ERR	BBK	UNC	IDNF	ABRT	TK0NF	AMNF
V	V		V				V		

PREREQUISITES - None.

DESCRIPTION - This command is used by the host to establish parameters which affect the execution of certain device features.

4.5.4.1 Transfer Modes

At power on, or after a hardware reset, the default setting of the ADR is PIO 4.

Set transfer mode:
A host can choose the transfer mechanism by Set Transfer Mode, subcommand code 03h, and specifying a value in the Sector Count register. The upper 5 bits define the type of transfer and the low order 3 bits encode the mode value.

Table 4-21: Transfer mode values

PIO default transfer mode	00000	000
PIO default transfer mode, disable IORDY	00000	001
PIO flow control transfer mode x	00001	nnn
Obsolete	00010	nnn
Multiword DMA mode x	00100	nnn
Reserved	01000	nnn
Reserved	10000	nnn
Key: nnn = a valid mode number in binary x = the mode number in decimal for the associated transfer type.		

4.5.5 Execute Drive Diagnostics

COMMAND CODE - 90h

PROTOCOL - Non-data.

INPUTS - None. The device selection bit in the Device/Head register is ignored.

Register	7	6	5	4	3	2	1	0
Features								
Sector Count								
Sector Number								
Cylinder Low								
Cylinder High								
Device/Head								
Command	90h							

NORMAL OUTPUTS - The diagnostic code written into the Error register is an 8-bit code. The possible error codes are listed in the next table.

Table 4-22: Diagnostic codes

Code	Description
01h	Device 0 passed, Device 1 passed or not present
00h, 02h-7Fh	Device 0 failed, Device 1 passed or not present
81h	Device 0 passed, Device 1 failed
80h, 82h-FFh	Device 0 failed, Device 1 failed

The meaning of values other than 01h and 81h are vendor specific and should be considered a diagnostic failed condition.

ERROR OUTPUTS - None. All error information is returned as a diagnostic code in the Error register.

Status register				Error register					
DRDY	DF	CORR	ERR	BBK	UNC	IDNF	ABRT	TK0NF	AMNF
V	V		V	The above table defines these values					

PREREQUISITES - None.

DESCRIPTION - This command shall perform the internal diagnostic tests implemented by the device. The DEV bit in the Device/Head register is ignored. Both devices, if present, shall execute this command.

5. Command Usage

5.1 ADR Characteristics

5.1.1 ADR Tape Positioning

The frame addresses start at zero from a predefined frame on the tape (at BOT for 15GB tapes and at COT for 25GB tapes), and increment by one for each tape frame in sequence.

5.1.2 Defect Management

Defects can occur infrequently on the tape. While some tapes may experience no write errors, most will experience some very small number of write errors. When a write error occurs, the application must manage the relocation of the data to new area further down the tape.
When reading the tape, the same location will result in a check condition on the read or the data will be from a previous write operation. In either case, the application must know where to look for the relocated data.
The example algorithms later in this section provide a mechanism for recovery from defects during writes and reads.

5.1.3 Write Data and Read Data Buffer Usage

For optimum bus utilization with the IDE interface drives, it is necessary for the application to monitor the buffer availability with Read Position commands. Because the drive does not support the ATA/ATAPI disconnect protocol nor the command overlap protocol, the drive will hold the bus until the buffer comes ready. The SCSI drives will perform a disconnect/reconnect protocol on Read and Write commands when the buffer is not available.

5.1.4 Prevent/Allow Media Removal

If the drive is in a state to prevent media removal, and the user presses the eject button, then the next command (other than Inquiry) shall generate a check condition for media removal prevented. It is important to note that under these circumstances, commands that return the check condition do not go to data phase. As an example, if the application was attempting to write data to the tape when the button was pushed, that data would need to be resent.

5.1.5 Host Command sequences

This section deals with tasks which can be performed sequentially without conflicting with the internal state of the ADR. The drive can be in one of the following states when it is performing media access commands :

- Reading
- Writing
- Locating

It is not possible to randomly sequence these states. The drive must first be in a Locating state before it can go into a Reading or Writing state. The drive can be put into a Locate state by means of the following commands Locate, Load and Rewind. The Load and Rewind commands position the tape to the first frame on tape.

Table 5-1: Allowed transitions from current to next command

Next Cur	READ	WRITE	LOCATE	REWIND	LOAD	UNLOAD
READ	Y	n	y	y	y	y
WRITE	N	y	y	y	y	y
LOCATE	Y	y	y	y	y	y
REWIND	Y	y	y	y	y	y
LOAD	Y	y	y	y	y	y
UNLOAD	N	n	n	n	y	n
MODE SELECT	N	n	y	y	y	y

Non media access commands can freely interrupt media access commands without altering the state of the drive. The only exception is the Mode Select Max Number of Retries page that influences settings needed for writing and reading.

5.2 Examples

5.2.1 Media Data Structures

The data structures defined in this document are provided as examples, they are provided in standard C.

5.2.1.1 Primitives

Primitives are provided for convenience and completeness, they may need modification depending on the host environment. It is important to remember that all data fields in the ADR logical format are stored in a big-endian format (MSB first, LSB last, in 8 bit bytes). Characters (char) are assumed to be 8 bit units.

```
typedef unsigned char BYTE8;  // 8 bit byte

typedef union _WORD16{
   // byte or word level access
   struct{
         BYTE8 msb;
         BYTE8 lsb;
   }b;
   unsigned short w;          // assumes 16 bit big-endian
} WORD16;                     // 16 bit word

typedef union _WORD32{
   // word or dword level access
   struct{
      WORD16 msw;
      WORD16 lsw;
   }w;
   unsigned long dw;       // assumes 32 bit big-endian
} WORD32;                     // 32 bit word

typedef struct _WORD64{
      WORD32 msdw;
      WORD32 lsdw;
} WORD64;                     // 64 bit word
```


5.2.1.2 AUX

The following structures provide the framework for the AUX data:

```
typedef struct _ADR_PARTITION
{
   BYTE8          partition_num;       // partition number
   BYTE8          par_desc_ver;        // version of this struct
   WORD16         wrt_pass_cntr;       // write pass counter
   WORD32         first_frame_addr;    // first frame in partition
   WORD32         last_frame_addr;     // last frame in partition
   WORD32         EOD_frame_addr;      // EOD addr in header frame
} ADR_PARTITION, *PADR_PARTITION;   // ADR partition description

// there are only two data access table entry flags combinations
// used in this model, one defines a single user data block, the
// other defines a filemark.
#DEFINE  ADR_DAT_FLAGS_DATABLK   0x0C  // BOLG=1, EOLG=1
#DEFINE  ADR_DAT_FLAGS_FILEMARK  0x01  // MARK=1

// data access table structures
typedef struct _ADR_DAT_ENTRY
{
   WORD32         blk_sz;  // 0x8000 for data (32KB), 0 for filemark
   WORD16         blk_cnt; // always 0x01 for now (1 blk/entry)
   BYTE8          flags;   // use one of the defines from above
   BYTE8          reserved;//
} ADR_DAT_ENTRY, *PADR_DAT_ENTRY; // ADR data access table entry

typedef struct _ADR_DAT
{
   BYTE8          dat_sz;        // sz of dat entries (0x08)
   BYTE8          reserved1;     //
   BYTE8          entry_cnt;     // active dat entries (0x01)
   BYTE8          reserved3;     //
   ADR_DAT_ENTRY  dat_list[16];  // logical group list
} ADR_DAT, *PADR_DAT;            // ADR data access table
```



```
typedef struct _ADR_AUX
{
   WORD32         format_id;           // hardware format rev (0)
   char           application_sig[4];  // ASCII text app ID string
   WORD32         hdwr;                // hdwr cntr,DO NOT USE
   WORD32         update_frame_cntr;   // use cntr for cnfg frames
   WORD16         frame_type;          // frame type descriptor
   BYTE8          reserved18_19[2];    //
   ADR_PARTITION  partition;           // partition description
   BYTE8          reserved36_43[8];    //
   WORD32         frame_seq_num;       // data frame sequence number
   WORD64         logical_blk_num;     // user data block number
   ADR_DAT        dat;                 // data access table
   BYTE8          reserved188_191[4];  //
   WORD32         filemark_cnt;        // filemarks before this blk
   WORD32         phys_fm[4];          // must be 0xFFFFFFFF filled
   WORD32         last_mark_addr;      // previous marker frame addr
   BYTE8          reserved204_223[20]; //
   BYTE8          app_specific[32];    // what ever you want it for
   BYTE8          reserved256_511[256];//
} ADR_AUX, *PADR_AUX;            // 512 byte ADR AUX data
```

The following defines are useful for the initialization examples that follow:

```
// frame type big-endian versions for word level access
#define ADR_FRAME_TYPE_HDR    0x0800   // header frame
#define ADR_FRAME_TYPE_DATA   0x8000   // data frame
#define ADR_FRAME_TYPE_MARK   0x0200   // filemark frame
#define ADR_FRAME_TYPE_EOD    0x0100   // eod frame
#define ADR_FRAME_TYPE_FILL   0x0000   // filler frame
```

The following initialization loads the AUX structure for writing the first header frame on a blank tape:

```
void InitHeaderAUX(PADR_AUX aux_ptr)
{
   // clear all so that reserved fields are all zero
   memset( (char *) aux_ptr, 0x00, 512);

   // the application signature will be loaded by hardware
   // once it has been initialized using Mode Select, so there
   // is no need to write it.
   // the update frame counter starts at zero, no need to change it

   // set the frame type
   aux_ptr->frame_type.w = ADR_FRAME_TYPE_HDR;

   // define the partition structure
   // config is not part of a data partition, it is identified as -1
   aux_ptr->partition.partition_num = 0xFF;
   // partition structure version number
```



```
    aux_ptr->partition.par_desc_ver = 0x01;
    // write pass counter doesn't mean anything for a random access
    // area on tape, the header frame is philosophically
    // random access, write pass counter is always 0xFFFF in config
    aux_ptr->partition.wrt_pass_cntr.w = 0xFFFF;
    // partitions are described by the first frame address and the
    // last frame address, config "partition" is an exception,
    // the config data is in two reserved sections, the first frame
    // for config is at 0x00000000 (and continuing for 20 frames
    // through 0x00000013), the second section is at 0x00000BA4 (and
    // continuing for 20 frames through 0x00000BB7). the application
    // must skipp over config frames when writing and reading user
    // data.
    aux_ptr->partition.first_frame_addr.dw = 0x00000000;
    aux_ptr->partition.last_frame_addr.dw = 0x00000BB7;

    // frame sequence number, logical block number, and dat
    // are not used in configuration frames leave zero filled

    // there are no filemarks in config
    aux_ptr->filemark_cnt.dw = 0;             //
    aux_ptr->phys_fm.dw = 0xFFFFFFFF;         // always 0xFFFFFFFF
    aux_ptr->last_mark_addr.dw = 0xFFFFFFFF; // indicates no fm

    // the following area is available for application specific
    // optimizations, please feel free to use it where it really
    // helps your product, please do not use it just because its
    // there. if somebody comes up with something really clever
    // maybe we can all adopt it.
    //aux_ptr->app_specific[32];
}
```


The following initialization loads the AUX structure for writing the first user data frame on a new tape:

```
void InitDataAUX(PADR_AUX aux_ptr)
{
   unsigned int segtrk, trks; // parameters for tape capacity
   unsigned long total_frames;

   // clear all so that reserved fields are all zero
   memset( (char *) aux_ptr, 0x00, 512);

   // the application signature will be loaded by hardware
   // once it has been initialized using Mode Select, so there
   // is no need to write it.

   // the update frame counter is not used outside config frames,
   // no need to change it.

   // set the frame type to EOD until we know better
   aux_ptr->frame_type.w = ADR_FRAME_TYPE_EOD;

   // define the partition structure
   // the data partition is identified as 0, so no need to write it
   // partition structure version number
   aux_ptr->partition.par_desc_ver = 0x01;
   // write pass counter starts at zero the first time a tape is
   // written, so its good
   // partitions are described by the first frame address and the
   // last frame address, the data partition starts at 0x00000014
   // and goes to the end of tape (remember that it must skip over
   // skipp over both sections of config frames when writing and
   // reading user data.
   aux_ptr->partition.first_frame_addr.dw = 0x00000014;

   // to determine the last frame on tape, the app must read Mode
   // page 0x2B to get the tape parameters.
   // you need to create something similar to what follows
   GetModePage2B(&segtrk, &trks);// get segtrk and trks fields
   if(segtrk == 19239 && trks == 24)
   {
      // we have the original tape with 15GB native capacity,
      // that means that it is a BOT (beginning-of-tape) load point
      // and we use the whole tape
      total_frames = segtrk * trks;
   }
   else
   {
      // we have a tape other than the original 15GB capacity,
      // that means that it is a COT (center-of-tape) load point
      // and we have to exclude 99 frames in the center for a
      // parking zone
      total_frames = (segtrk-99) * trks;
   }
   // we now know the size of the tape
   aux_ptr->partition.last_frame_addr.dw = total_frames;

   // frame sequence number, logical block number, start at zero.
   // in the example BuildDataBlkAUX, BuildFilemarkAUX, and
```



```
    // BuildEODAUX routine below, we increment these counters as
    // required, so for this example well start them at –1 here.
    aux_ptr->partition. frame_seq_num.dw = 0xFFFFFFFF;
    // we are cheating here a little, with 32KB logical
    // block we will never saturate the least significant
    // dword in the 64 bit logical block number field
    // before we hit EOD, so we are ignoring the high dword
    aux_ptr->partition.logcal_blk_num.lsdw = 0xFFFFFFFF;

    // the dat will always have one entry, either a data block or a
    // filemark, but for now we don't know which it will be, but we
    // can fill in some of the constant fields.
    aux_ptr->dat.dat_sz = 0x08;

    // there are no filemarks yet
    aux_ptr->filemark_cnt.dw = 0;            //
    aux_ptr->phys_fm.dw = 0xFFFFFFFF;        // always 0xFFFFFFFF
    aux_ptr->last_mark_addr.dw = 0xFFFFFFFF; // indicates no fm

    // the following area is available for application specific
    // optimizations, please feel free to use it where it really
    // helps your product, please do not use it just because its
    // there. if somebody comes up with something really clever
    // maybe we can all adopt it.
    //aux_ptr->app_specific[32];
}
```

The following is an example of how a data frame might be written. It assumes the AUX buffer pointed to by `aux_ptr` was initialized and is currently a static variable allocated by the calling function. This is a simplified write queuing routine and does not address error handling, the intent is to show which fields to modify in AUX during writing.

```
void BuildDataBlkAUX(PADR_AUX aux_ptr)
{
    // set the frame type to data
    aux_ptr->frame_type.w = ADR_FRAME_TYPE_DATA;

    aux_ptr->frame_seq_num.dw++;
    // we are cheating here a little, with 32KB logical
    // block we will never saturate the least significant
    // dword in the 64 bit logical block number field
    // before we hit EOD, so we are ignoring the carry to
    // the high dword
    aux_ptr->logcal_blk_num.lsdw++;

    // this model is always one dat entry and a single 32KB user data
    // logical block
    aux_ptr->dat.entry_cnt = 1;
    aux_ptr->dat.dat_list[0].blk_sz.dw = 0x8000;   // 32KB blk
    aux_ptr->dat.dat_list[0].blk_cnt.w = 1;  // 1 blk in this frame
    // logcal blk group starts and ends this frame
    aux_ptr->dat.dat_list[0].flags = ADR_DAT_FLAGS_DATABLK;
}
```

The following is an example of how a filemark frame might be written. It assumes the AUX buffer pointed to by `aux_ptr` was initialized and is currently a static variable allocated by the calling function. This is a

simplified write queuing routine and does not address error handling, the intent is to show which fields to modify in AUX during writing.

```
void BuildFilemarkAUX(PADR_AUX aux_ptr)
{
   // set the frame type to data
   aux_ptr->frame_type.w = ADR_FRAME_TYPE_MARK;

   aux_ptr->frame_seq_num.dw++;
   // we are cheating here a little, with 32KB logical
   // block we will never saturate the least significant
   // dword in the 64 bit logical block number field
   // before we hit EOD, so we are ignoring the carry to
   // the high dword
   aux_ptr->logcal_blk_num.lsdw++;

   // this model is always one dat entry and a single filemark
   // logical block
   aux_ptr->dat.entry_cnt = 1;
   aux_ptr->dat.dat_list[0].blk_sz.dw = 0x0;//
   aux_ptr->dat.dat_list[0].blk_cnt.w = 1;  // 1 blk in this frame
   // logcal element is a filemark
   aux_ptr->dat.dat_list[0].flags = ADR_DAT_FLAGS_FILEMARK;
}

// we need to show writing for filemarks because there are
// fields that need to be modified for AUX after the data
// is sent to tape
void WriteADRFilemark(PADR_AUX aux_ptr, LPBYTE buffer)
{
   DWORD frame_addr;

   BuildFilemarkAUX(aux_ptr);

   // record where we are
   frame_addr = ReadPosition();

   // send the 32.5KB data + AUX example call
   WriteTape(buffer, aux_ptr);

   // update the file tracking fields for later frames
   aux_ptr->filemark_cnt.dw++;
   aux_ptr->last_mark_addr.dw = frame_addr;
}
```

The following is an example of how an EOD frame might be written. It assumes the AUX buffer pointed to by `aux_ptr` was initialized and is currently a static variable allocated by the calling function. This is a simplified write queuing routine and does not address error handling, the intent is to show which fields to modify in AUX during writing.

```
void BuildEODAUX(PADR_AUX aux_ptr)
{
   // set the frame type to data
   aux_ptr->frame_type.w = ADR_FRAME_TYPE_EOD;
```



```
   aux_ptr->frame_seq_num.dw++;

   aux_ptr->dat.entry_cnt = 0;
}
```

The following routine creates a filler frame:

```
void CreateFillerFrame(PADR_AUX aux_ptr)
{
   // clear all so that reserved fields are all zero,
   // this also results in a frame type of zero which
   // is a filler frame.
   memset( (char *) aux_ptr, 0x00, 512);
}
```

5.2.1.3 Header Frame

The following data structures are used for the data space of the header frames:

```
typedef struct _ADR_PAR_LIST
{
   BYTE8          par_num; // number of partitions in list (0x01)
   BYTE8          reserved1_3[3];   //
   ADR_PARTITION  partition[16];    // partition descriptions
} ADR_PAR_LIST, *PADR_PAR_LIST;     // ADR partition list structure

typedef struct _ADR_HEADER
{
   char           ident_str[8];  // "ADR_SEQ" seq access model
   BYTE8          major_rev;     // development doc major rev(0x01)
   BYTE8          minor_rev;     // development doc minor rev(0x01)
   BYTE8          reserved10_15[6]; //
   ADR_PAR_LIST   par_list;      // partition list
} ADR_HEADER, *PADR_HEADER;            // ADR header frame structure
```

The following initialization loads the header structure for a new tape:

```
void CreateHeaderFrame(PADR_HEADER buffer)
{
   unsigned int segtrk, trks; // parameters for tape capacity
   unsigned long total_frames;

   // clear all so that reserved fields are all zero
   memset( (char *) buffer, 0x00, 0x8000);  // clear 32K buffer

   strcpy(buffer->ident_str, "ADR_SEQ");
   buffer->major_rev = 1;
   buffer->minor_rev = 1;

   // the configuration area is not a true partition and is not
   // recorded, the only partition defined in this model is user
   // data partition 0.
   buffer->par_list.par_num = 1;
   buffer->par_list.partition[0].partition_num = 0;
   buffer->par_list.partition[0].par_desc_ver = 1;
```



```
      // write pass counter starts at zero
      // buffer->par_list.partition[0].wrt_pass_cntr = 0;


      // determining the partition parameters is the same as
      // the AUX initialization for data frames
      aux_ptr->partition.first_frame_addr.dw = 0x00000014;
      // to determine the last frame on tape, the app must read Mode
      // page 0x2B to get the tape parameters.
      // you need to create something similar to what follows
      GetModePage2B(&segtrk, &trks);// get segtrk and trks fields
      if(segtrk == 19239 && trks == 24)
      {
         // we have the original tape with 15GB native capacity,
         // that means that it is a BOT (beginning-of-tape) load point
         // and we use the whole tape
         total_frames = segtrk * trks;
      }
      else
      {
         // we have a tape other than the original 15GB capacity,
         // that means that it is a COT (center-of-tape) load point
         // and we have to exclude 99 frames in the center for a
         // parking zone
         total_frames = (segtrk-99) * trks;
      }
      // we now know the size of the tape
      aux_ptr->partition.last_frame_addr.dw = total_frames;
   }
```

5.2.2 Mounting a tape

When mounting the tape the application must first determine if the tape has been previously written or if it is blank from manufacture. Coming from manufacturing a tape may have some frames with data written in them, but this data will be related to test patterns and will have no meaningful content. New tapes may also have only the embedded servo information, in which case there are no frames recorded on the tape. If there is understandable data on the tape, then the application must first read the configuration frames and determine what the nature of the data is. Be advised that there are also early applications that do not use this standard. If a tape has readable data on it, it is best warn the user that old data may be lost before initializing a tape whenever possible.

5.2.3 Initializing a tape

If the tape has not been previously used, then the application must initialize the tape with basic control structures. When initializing a tape the application must create the configuration frames in the reserved locations to indicate the specifics of the user data partitions.

5.2.4 Writing from BOP

If the tape has been previously been written, the application must increment the write pass counter in the header frame partition list. The application must initialize the data frame AUX to the new write pass counter, and it must indicate that the EOD is now at BOP until data has been written.

5.2.5 Appending on a tape

Whenever appending to the user data partition the application must read the header frames and locate to the EOD frame listed for partition 0. From the EOD frame the application can initialize the AUX for the append operation. When writing commences, the application must overwrite the EOD frame, replacing it with new user data. When the append is complete the application must flush buffers to tape then locate to the header positions and update the header tables with the new EOD location.

5.2.6 Writing a tape with defect management

The following sequence is an example used to write the tape in a sequential fashion. The application must manage tape defects by relocating when a write error is encountered. Due to the drive's data buffer, write errors are deferred, the error actually occurs as the data is being written from the drive buffer onto the tape. If an error is encountered, the application must resubmit all of the data beginning from the frame that experienced the error. Many optimizations are possible, including using the read position data to determine the number of consecutive writes that can be issued.

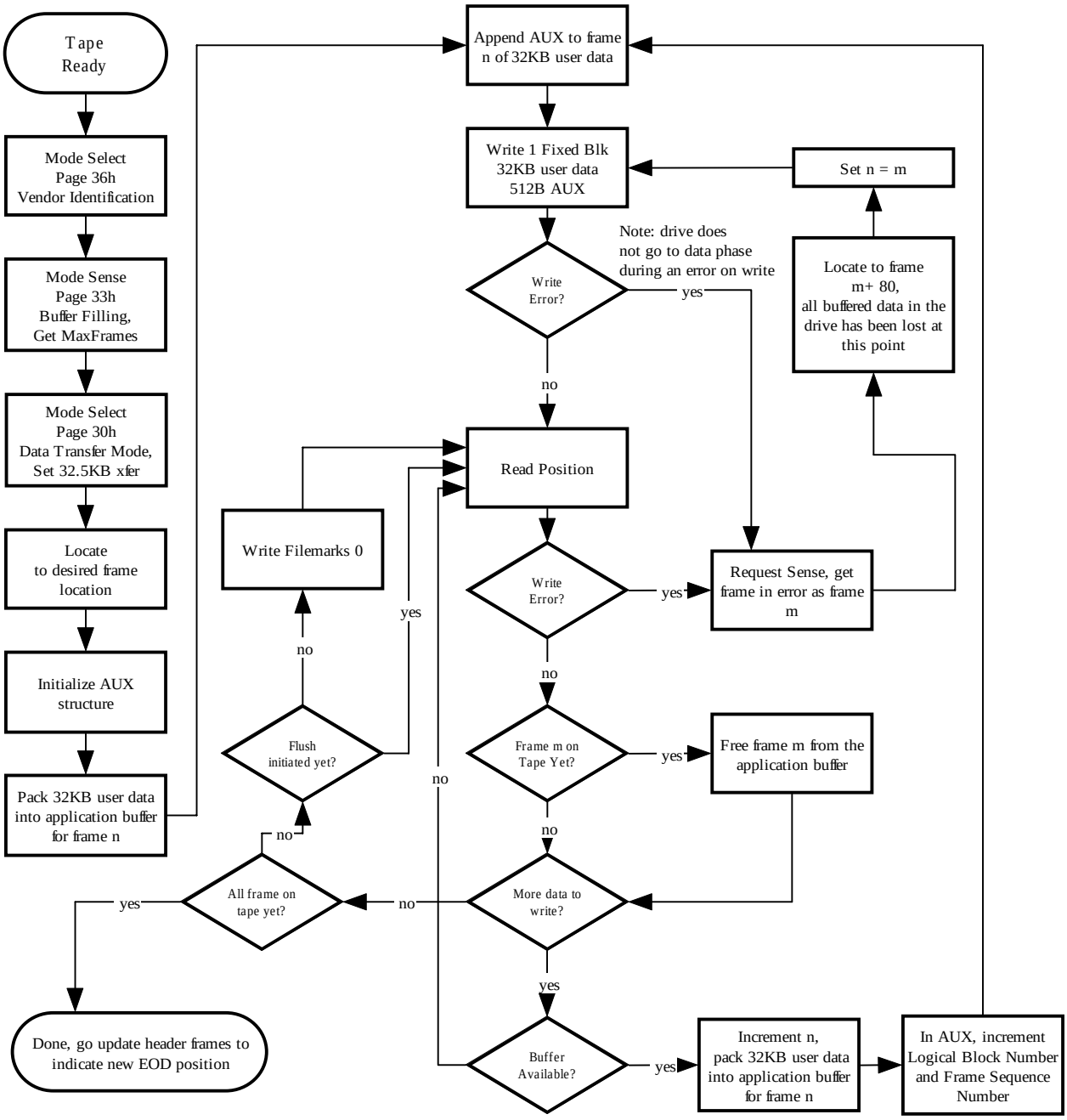

Figure 5-1 Example write algorithm

5.2.7 Reading a tape with defect management

The following sequence is used to read the tape in a sequential fashion. The read algorithm needs to accommodate up to 10 consecutive read errors (while this should never occur it is the same threshold used in write operations. Many optimizations are possible, including using the read position data to determine the number of consecutive reads that can be issued.

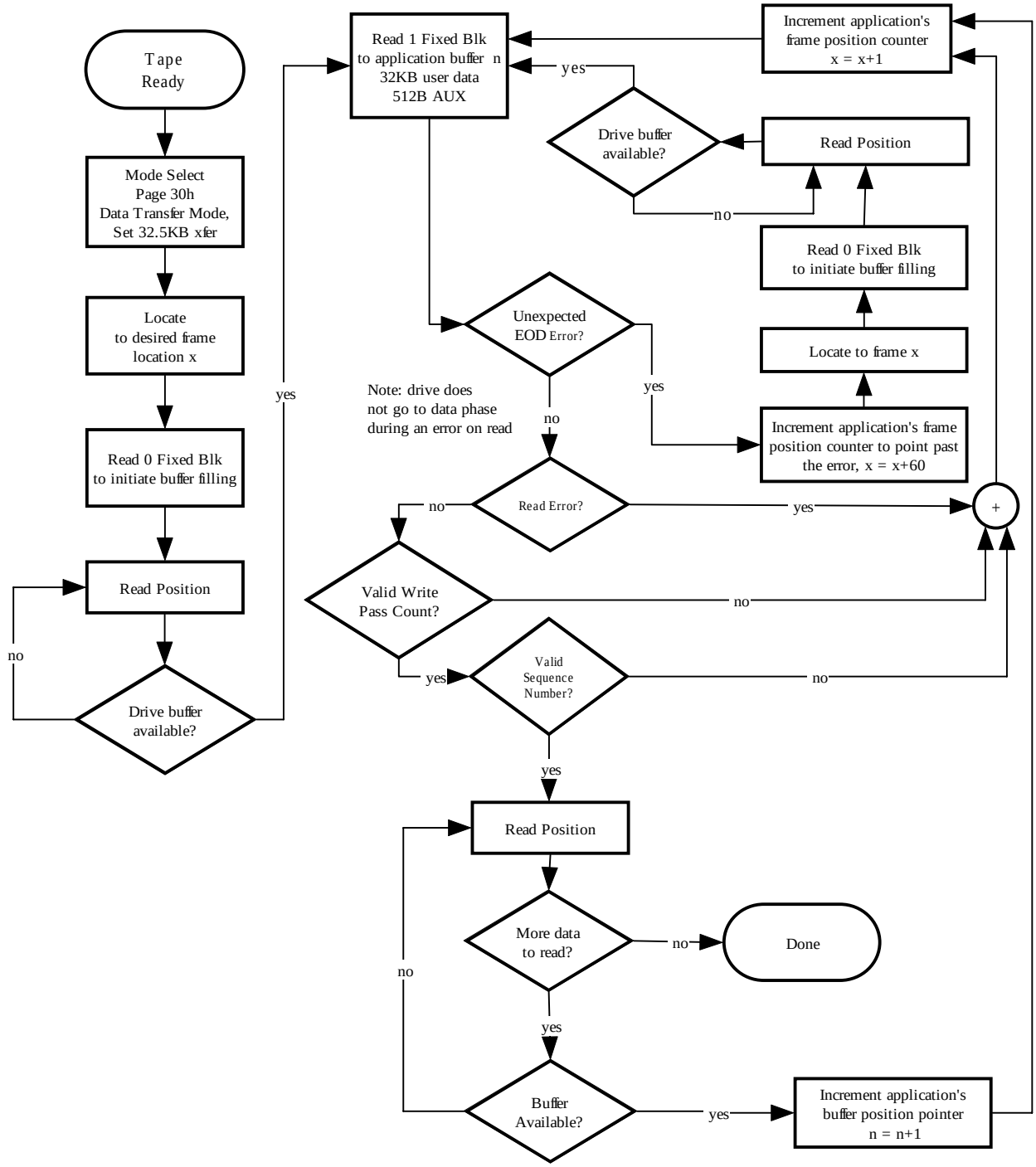

Figure 5-2 Example read algorithm

